THE UNIVERSITY *of* EDINBURGH

Brain Perfusion Imaging
- Performance and Accuracy

Fan Zhu

Doctor of Philosophy
Centre for Intelligent Systems and their Applications
School of Informatics
University of Edinburgh
2012

Abstract

Brain perfusion weighted images acquired using dynamic contrast studies have an important clinical role in acute stroke diagnosis and treatment decisions. The purpose of my PhD research is to develop novel methodologies for improving the efficiency and quality of brain perfusion-imaging analysis so that clinical decisions can be made more accurately and in a shorter time. This thesis consists of three parts:

- My research investigates the possibility that parallel computing brings to make perfusion-imaging analysis faster in order to deliver results that are used in stroke diagnosis earlier. Brain perfusion analysis using local Arterial Input Functions (AIF) techniques takes a long time to execute due to its heavy computational load. As time is vitally important in the case of acute stroke, reducing analysis time and therefore diagnosis time can reduce the number of brain cells damaged and improve the chances for patient recovery. We present the implementation of a deconvolution algorithm for brain perfusion quantification on GPGPU (General Purpose computing on Graphics Processing Units) using the CUDA programming model. Our method aims to accelerate the process without any quality loss.

- Specific features of perfusion source images are also used to reduce noise impact, which consequently improves the accuracy of hemodynamic maps. The majority of existing approaches for denoising CT images are optimized for 3D (spatial) information, including spatial decimation (spatially weighted mean filters) and techniques based on wavelet and curvelet transforms. However, perfusion imaging data is 4D as it also contains temporal information. Our approach using Gaussian process regression (GPR) makes use of the temporal information in the perfusion source imges to reduce the noise level. Over the entire image, our noise reduction method based on Gaussian process regression gains a 99% contrast-to-noise ratio improvement over the raw image and also improves the quality of hemodynamic maps, allowing a better identification of edges and detailed information. At the level of individual voxels, GPR provides a stable baseline, helps identify key parameters from tissue time-concentration curves and reduces the oscillations in the curves. Furthermore, the results show that GPR is superior to the alternative techniques compared in this study.

- My research also explores automatic segmentation of perfusion images into po-

tentially healthy areas and lesion areas, which can be used as additional information that assists in clinical diagnosis. Since perfusion source images contain more information than hemodynamic maps, good utilisation of source images leads to better understanding than the hemodynamic maps alone. Correlation coefficient tests are used to measure the similarities between the expected tissue time-concentration curves (from reference tissue) and the measured time-concentration curves (from target tissue). This information is then used to distinguish tissues at risk and dead tissues from healthy tissues. A correlation coefficient based signal analysis method that directly spots suspected lesion areas from perfusion source images is presented. Our method delivers a clear automatic segmentation of healthy tissue, tissue at risk and dead tissue. From our segmentation maps, it is easier to identify lesion boundaries than using traditional hemodynamic maps.

Acknowledgements

I would like to express my sincerest gratitude to my supervisors, Prof. Malcolm Atkinson, Dr. David Rodriguez Gonzalez, Dr. Trevor Carpenter and Prof. Joanna Wardlaw. They have given their time generously, been a great sources of guidance, support, inspiration and encouragement. They have led me into inter-disciplinary research and taught me principles of research, how to generate and implement ideas. Without their help, this thesis would never have happened. My first supervisor, Malcolm, gave me 48 pages of hand written comments including 203 open questions, which aided the thesis substantially, not to mention his brilliant advice and extraordinary supervision.

Many thanks go to Dr. Jano van Hemert and Dr. Paolo Besana also. They have supervised me for one year and gave me invaluable help and guidance.

This project is supported by the SINAPSE (Scottish Imaging Network - A Platform for Scientific Excellence) collaboration (www.sinapse.ac.uk) funded by the Scottish Funding Council and the Chief Scientist Office. This project is also funded by the Scottish Overseas Research Student Awards Scheme (SORSAS). I am grateful to the SINAPSE and the SORSAS for their funding for my study.

The work was carried out at the Brain Research Imaging Centre (BRIC), Neuroimaging Sciences, Edinburgh (www.bric.ed.ac.uk), a core area of Wellcome Trust Clinical Research Facility and part of the SINAPSE collaboration.

This work has leveraged the resources provided by Edinburgh Compute and Data Facility (ECDF) (www.ecdf.ed.ac.uk) and School of Informatics, University of Edinburgh.

All patient data came from Multi-Centre Acute Stroke Imaging Study which was funded by the Translational Medicine Research Collaboration (TMRC).

Finally, I would like to thank my beloved wife, Weiting Zhang, for her support and understanding during my PhD study. I am also always inspired by her lovely face ☺. I would like to thank my parents and my friends, for their encouragement.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Fan Zhu*)

Table of Contents

List of Abbreviations

AIF - Arterial Input Function

CBF - Cerebral Blood Flow

CBV - Cerebral Blood Volume

CNR - Contrast-to-Noise Ratio

CPU - Central Processing Unit

CT - Computed Tomography

GPU - Graphics Processing Unit

GPGPU - General Purpose computing on Graphics Processing Units

GPR - Gaussian Process Regression

IRF - Impulse Response Function

MGPR - Multiple observations Gaussian Process Regression

MRI - Magnetic Resonance Imaging

MTT - Mean Transit Time

PCT - Perfusion Computed Tomography

SVD - Singular Value Decomposition

TIPS - Time-Intensity Profile Similarity

Tmax - Time to maximum

Publications directly related to results presented in this thesis

F. Zhu, D. Rodriguez Gonzalez, T. Carpenter, M.P. Atkinson, and J. Wardlaw, 'A parallel deconvolution algorithm in perfusion imaging' in 2011 First IEEE International Conference on Healthcare Informatics, Imaging and Systems Biology (HISB), pp. 278-283, IEEE, 2011.

F. Zhu, T. Carpenter, D. Rodriguez Gonzalez, M.P. Atkinson, and J. Wardlaw, 'Computed tomography perfusion imaging denoising using Gaussian process regression' in Physics in Medicine and Biology, vol. 57, no. 12, pp. N183-198, 2012.

F. Zhu, D. Rodriguez Gonzalez, T. Carpenter, M.P. Atkinson, and J. Wardlaw, 'Parallel Perfusion Imaging Processing Using GPGPU' in Computer Methods and Programs in Biomedicine, 2012.

F. Zhu, D. Rodriguez Gonzalez, T. Carpenter, M.P. Atkinson, and J. Wardlaw, 'Automatic Lesion Area Detection Using Source Image Correlation Coefficient for CT Perfusion Imaging' in the IEEE Transactions on Information Technology in Biomedicine. (Under Revision)

Chapter 1

Introduction

1.1 Aims of chapter

This thesis contains interdisciplinary research that combines concepts from informatics and medical science. To make this thesis accessible to audiences with different backgrounds, this chapter describes notions and approaches for brain perfusion imaging and computer science technologies. The reader may want to skip sections covering background he or she is familiar with.

Section 1.4 discusses perfusion imaging acquisition and follow up processing technologies. Section 1.5 introduces the notions of brain hemodynamic parameters. Sections 1.6.1, 1.6.2 and 1.7 present how CT and MRI imaging works and their workflows. Section 1.8 covers deconvolution in perfusion-imaging analysis. Section 1.9 talks about local AIF technique. Section 1.10 introduces the features of lesion vision. Section 1.11 illustrates the importance of time in stroke treatments. Section 1.13 presents the goal and synopsis of this thesis and Section 1.14 states the outline of the subsequent chapters.

1.2 Brain Perfusion Imaging

Perfusion is defined as the passage of fluid through the blood vessels or lymphatic system to an organ or a tissue. Scanning using perfusion, also known as perfusion imaging, is able to observe, record and quantify the perfusion in a human body. The perfusion information is important and is used to ascertain data on the blood flow to vital organs such as the heart and brain. Brain perfusion weighted images acquired using dynamic contrast studies have an important clinical role in acute stroke diagnosis

and treatment decisions [1].

Numerous techniques have been proposed in the past to measure various perfusion related parameters in the brain. The main imaging techniques used to evaluate brain hemodynamic parameters are dynamic perfusion computed tomography (PCT), MRI dynamic susceptibility contrast (DSC), positron emission tomography (PET), single photon emission computed tomography (SPECT), Xenon-enhanced computed tomography (XeCT), arterial spin labeling (ASL), and Doppler ultrasound. These techniques for assessment of tissue perfusion or blood flow are important for clinical areas such as ischemic diseases, cancer and tumors. In the case of acute diseases, people who have had a stroke will need a brain scan, blood tests, blood pressure checks and an electrocardiogram (ECG) to find the cause of the stroke, what damage it has done and what medical treatment is needed. The information obtained from the brain scan can be used to evaluate the appropriateness of administering thrombolytic treatment, which can help to reduce the final volume of dead tissues. In the case of cancers, brain scans can be used to monitor cancer therapies. In the case of tumors, these imaging techniques are used to distinguish tumor characteristics and follow tumor development, possibly also after treatment to see whether it has been effective.

1.3 Stroke

A stroke, one of the three most common fatal health problems in the world[1], is a serious medical condition. Most strokes are due to a cut off of the blood supply to a part of the brain. When the brain loses its energy supply carried by blood, the brain functions are rapidly lost, severe damage to brain tissues are caused and a stroke is resulted in. Stroke affects a lot of people and the damage is serious.

Based on the investigation of the World Health Organisation, 15 million people worldwide suffer a stroke annually. A third of them die and another third of them survive with permanent disability [2]. The Stroke Association [3], a UK-wide charity solely concerned with stroke, states that stroke is the leading cause of severe disability with 350,000 people affected at any one time in UK. Stroke is also the largest cause of adult disability in the United States and Australia based on the American Stroke Association[2] [4] and the National Stroke Foundation of Australia [5].

[1] The other two most fatal disorders are heart disease and cancer.

[2] Every 40 seconds, someone in America suffers a stroke. It's the fourth leading cause of death and a leading cause of disability.

Stroke symptoms can vary, depending on the area of the brain affected. They include unilateral paralysis, and problems with speech and the damage caused by stroke can be neurological or fatal.

Most strokes can be classified into three categories:

- **Ischemic stroke** This type of stroke accounts for more than four-fifth of all cases. It is mostly caused by thrombosis and embolism. Thrombosis means blood supply is interrupted by a blood clot forming inside a blocked blood vessel. Embolic stroke is the result of clots coming from anywhere else in the body but then travelling to the brain and causing an obstruction. Ischemic stroke can be treated using alteplase, a medicine which dissolves the clot. An anti-platelet medicine may also be used to make the platelets less sticky, which reduces the chances of further blood clots occurring.

- **Hemorrhagic stroke** This type of stroke occurs when a blood vessel bursts or breaks, causing bleeding and damage in the brain. Hemorrhagic stroke can be treated by surgery, which removes the hematoma and stops the damaged blood vessels from bleeding again. Medicines to lower blood pressure may also be given to the patient.

- **Transient Ischemic Attack (TIA)** Also known as a mini stroke. This type of stroke is caused by a temporary clot. It is a warning of a further stroke that should be taken seriously. Its symptoms are the same as for a normal stroke but will resolve within a few minutes or a maximum of 24 hours.

1.3.1 Infarct Core and Penumbra Area

Infarct core is at the centre of ischemic stroke, where the blood supply is most badly affected and tissues in an infarct core are irreversibly damaged and usually dead within a few minutes. The penumbra area is the area around an infarct core. The penumbra tissue is also called “tissues at risk” and is affected by the stroke but still has the potential to be salvaged with treatment. Tissues in both the infarct core and penumbra area are also called abnormal tissues and can be observed using medical imaging methods.

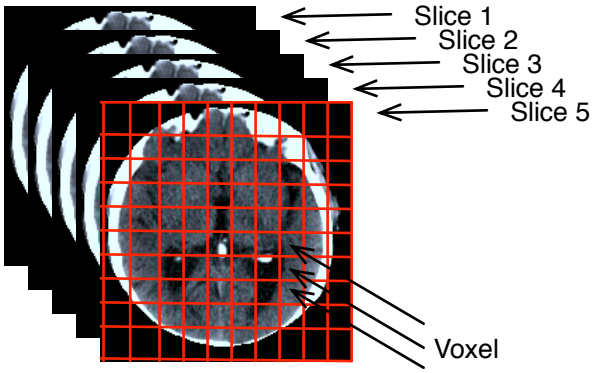


Figure 1.1: Slices and Voxels
This figure illustrates the structure of a CT image.

1.4 Image Acquisition

A CT or MRI image is a 3D image volume consisting of several 2D images where each 2D image is typically called a slice, because it corresponds to a certain thickness of the object being scanned. Furthermore, each 2D slice is composed of many elements, each of which represents an area with a certain thickness. As a result, the term 'voxel' is introduced to stand for the element. In other words, an image slice consists of voxels, whereas a typical digital image is composed of pixels. Figure 1.1 is an example of a CT image.

1.4.1 CT Imaging Acquisition

In CT images, the grey levels reflect the proportion of X-rays scattered or absorbed as they pass though each voxel. A CT image is created by directing X-rays from multiple orientations and measuring their attenuation in intensity. Reconstruction is then used to convert the X-ray attenuation into a density distribution within the slices. Furthermore, by acquiring a contiguous series of CT image slices, data describing an entire volume of the brain can be obtained.

CT imaging provides clear and detailed images, not only of muscle and soft tissue, but also of bones and blood vessels. However, there are risks associated with the use of CT scanning. The two main risks are: 1) CT scanning increases the risk of cancer due to the exposure of X-ray radiation during the scan. 2) The contrast agent injected to the patient may lead to allergic reactions or kidney failure.

1.4.2 CT Imaging Reconstruction

Reconstruction is a mathematical process that converts sinograms, a visual representation of the raw data obtained in tomography scans, into multi-dimensional slice images. Two of the most widespread reconstruction techniques are filtered back projection [6] and Shepp-Logan filter [7].

In the image reconstruction, tissues are subdivided into individual tissue voxels. All of the structures within an individual voxel are mixed together and represented by a single CT number. No details can be discriminated within a voxel. When looking at images, voxels are seen side by side. As a result, in order to achieve highly detailed images, it is necessary to use small voxels [8].

1.4.3 MRI Imaging Acquisition

The principle behind MRI imaging is that it makes use of the property of nuclear magnetic resonance to image nuclei of atoms inside the body. In 1971, Raymond Damadian reported that tumors and normal tissue can be distinguished by a nuclear magnetic resonance (MR) scan [9]. He also suggested that this information could be used to diagnose cancer. MRI imaging has become a powerful clinical tool for the evaluation of brain anatomy. It can be used to measure a number of functional or metabolic parameters.

MRI images are commonly acquired from either spin-echo or gradient-echo sequences. On a T_1-weighted scan, water-containing tissues appear dark and fat-containing tissues appear bright. Opposite to a T_1-weighted scan, on a T_2-weighted scan, water appears bright and fat appears dark.

An MRI image is usually reconstructed using 2D Fourier transform.

1.4.4 CT and MRI Imaging Resolution

Each volume of brain CT or MRI scan images usually has a spatial resolution of 1-2 millimetres and the acquired data can have a resolution from (128,128,2) to (512,512,16) voxels. Brain hemodynamic quantities need to be computed for each voxel. The spatial resolution of a perfusion image is the same as the resolution of a non-perfusion one, but a perfusion study has multiple 3D volumes acquired at different sampling time points while a non-perfusion one only has one 3D volume.

1.5 Brain Hemodynamics

Dynamic perfusion computed tomography (CT perfusion imaging) and MRI dynamic susceptibility contrast (MRI perfusion imaging) provide measurements of brain hemodynamic quantities, such as cerebral blood flow (CBF), cerebral blood volume (CBV) and mean transit time (MTT), which have an important clinical role in acute stroke diagnosis and treatment decisions. In most perfusion methods, results are evaluated based on the symmetrical features between the two hemispheres and the experience of clinical experts. Furthermore, the analysis of perfusion imaging requires serial analysis of arterial input.

Both the CT and MRI perfusion imaging techniques request an injection of a contrast agent. Contrast agents are used to improve the visibility of blood vessels and internal body structures. CT perfusion imaging commonly uses iodine-based contrast material while MRI perfusion imaging often uses gadolinium-based contrast material.

Hemodynamic parameters introduced in this section are all voxel specific. For any $Parameter_i$ (CBF, CBV, MTT or Tmax, etc.), it is voxel based and refers to the hemodynamic parameter for $voxel_i$, where $voxel_i$ is in some position (x_i, y_i, z_i) in the entire image. Furthermore, equations mentioned in the subsequent sub sections are also voxel based.

1.5.1 CBF Parameter

The concept of cerebral blood flow (CBF) was proposed by Seymour S. Kety in 1945 [10]. He developed a method to quantitatively determine cerebral blood flow by means of arterial and internal jugular blood concentrations of an inert gas during the first ten minutes of its inhalation at low concentration.

In the article by Lassen in 1959 [11], the early measurement methods of CBF were reviewed and the concept of auto regulation of cerebral blood flow was presented in detail for the first time. He stated that in normal individuals, not given pharmacological agents, the cerebral oxygen uptake is remarkably constant during sleep, physical or intellectual effort. The cerebral oxygen consumption is also not affected by relatively marked changes of the arterial blood pressure and gas tensions of arterial blood. We also established that the cerebral oxygen uptake is subnormal only in acute and chronic cerebral disorders characterised by distinct signs of mental hypofunction, depression of consciousness in acute disorders, and loss of intellectual faculties in chronic disorders.

1.5.2 Residue Function and Impulse Response Function

To describe the tissue retention of a tracer, the residue function $R(t)$ is introduced. $R(t)$ presents the fraction of the tracer present in tissue at the t_{th} sampling time point. Consequently, the $R(t)$ reaches its maximal value, 1, after the injection of the contrast agent and decreases to 0 when time passes long enough ($R(\infty) = 0$).

The tissue time-concentration function C is

$$C(t_i) = \int_0^{t_i} C_a(\gamma) \cdot R(T-\gamma) \mathrm{d}\gamma \quad \cdot CBF \tag{1.1}$$

where t_i is the sampling time point and T is the time interval, $C(t)$ is the tissue time-concentration function and $C_a(t)$ is the artery time-concentration function, which is known as artery input function (AIF).

The product, $R(t) \cdot CBF$, is called the tissue Impulse Response Function (IRF), since it can be considered as a result of the aforementioned impulse with infinitesimal input. For every voxel, there is a CBF value and a residue function which form the IRF together. $R(t)$ can be considered as a relative function while IRF is an absolute function.

As the artery time-concentration function $C_a(t)$ is variable in time, the tissue time-concentration time $C(t)$ becomes the convolution of the impulse response function and the AIF [12], as:

$$C(t) = AIF \otimes R(t) \cdot CBF \qquad or \qquad C(t) = AIF \otimes IRF \tag{1.2}$$

In practice, the AIF is obtained from a major artery, such as the middle cerebral or intracranial internal carotid artery with two assumptions: (a) it is the only input to the tissue of interest; (b) there is no circulatory or dispersion delay of contrast material [13]. A poor selection of the AIF may lead to an incorrect result. For example, if an AIF is obtained from an obstructed vessel, the signal arrival time (time of arrival of contrast agent bolus) in the rest of the brain can precede the arrival time in the AIF.

1.5.3 CBV Parameter

Cerebral blood volume (CBV) is as crucial as cerebral blood flow and is also a useful measurement that can be obtained easily from the same data used for determining cerebral blood flow. In a review article in 1990, Rosen *et al.* [14] stated that, by analyzing tissue time-concentration curves while dynamically tracking the passage of a bolus of a high-susceptibility contrast agent, maps of CBV can be calculated. This technique

can be used in methods that track the passage of a contrast agent with high temporal resolution, such as dynamic perfusion computed tomography and MRI dynamic susceptibility contrast.

CBV can be determined from the ratio of the areas under the tissue's and arterial's time-concentration curves, as

$$CBV = \frac{\int_{-\infty}^{\infty} C(t)\mathrm{d}t}{\int_{-\infty}^{\infty} C_a(t)\mathrm{d}t} \tag{1.3}$$

where $C(t)$ and $C_a(t)$ are the intensity values for the target tissue and artery, respectively.

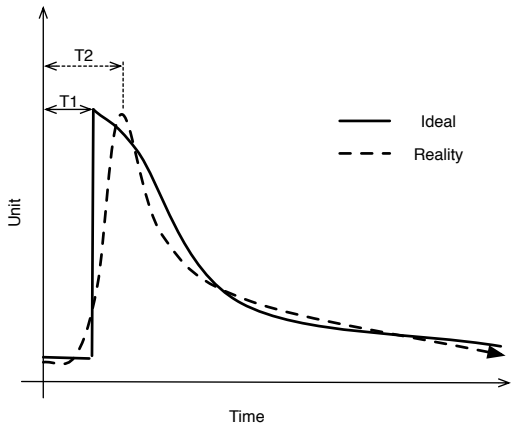


Figure 1.2: The Measurement of Tmax

This figure indicates how Tmax is determined. The solid line indicates the ideal intensity function ($R(t)$) and the dashed line corresponds to the intensity function that is measured.

1.5.4 Tmax Parameter

The Tmax represents the time from the start of the scan until the maximum intensity of contrast material arrives at each voxel. Figure 1.2 illustrates the determination of Tmax. *T1* represents the expected Tmax for the ideal situation. However, in reality, intensity cannot rise to its peak instantaneously so there will be a delay of the maximum intensity value occurring. As a result, the measured Tmax value, *T2* in the figure, is larger than its theoretic value, *T1*. Tmax has been recently used in clinical trials as one of the criteria for judging the abnormality of brain perfusion [15]. Tmax maps are usually used to estimate the blood flow compromise (which is anything that can prevent proper blood flow) and the penumbra area.

1.5.5 MTT Parameter

Mean transit time, which is known as MTT, is derived from the distribution of transit times and as the name suggests, it refers to the average time it takes the blood to pass through a given region of tissue. The rCBF, rCBV and rMTT denotes relative CBF, CBV and MTT. The parameters rCBF, rCBV and rMTT are relative measurements instead of absolute, as:

$$CBV = CBF \times MTT \tag{1.4}$$

$$rCBF = \frac{\text{net blood flow through a voxel}}{\text{mass of the voxel}} \tag{1.5}$$

$$rCBV = \frac{\text{volume of blood in a voxel}}{\text{mass of the voxel}} \tag{1.6}$$

$$rMTT = \frac{rCBV}{rCBF} \tag{1.7}$$

Equation 1.4 is called the Central Volume Principle, which denotes the relationship between CBF, CBV and MTT.

1.6 Brain Perfusion Imaging

Perfusion imaging refers to techniques used to non-invasively measure how blood travels though the vasculature, via assessment of cerebral hemodynamic perfusion parameters such as CBF, CBV and MTT. These techniques can be used as clinical tools in diagnosis and treatment of patients with cerebrovascular disease and other brain disorders including the evaluation of tissue at risk after acute stroke, non-invasive histologic assessment of tumors, evaluation of neurodegenerative conditions such as Alzheimer's disease and the effects of drugs used to treat those conditions [16].

To generate the hemodynamic parameters as final output, the tissue time-concentration curve and the Arterial Input Function (AIF) need to be determined from the source images.[3] An AIF is a tissue time-concentration curve of an artery or an average tissue time-concentration curve of several artery tissues. Thus the artery needs to be identified before the determination of the AIF. In this thesis, AIFs used in the experiment are obtained using a software called *Perfusion Mismatch Analyzer* [17].

The deconvolution of the tissue time-concentration curve and the AIF yields the tissue Impulse Response Function (IRF), which is also known as the tissue response

[3]The tissue time-concentration curve is determined for each voxel; the AIF can be determined either voxel based or scan based depending on the AIF selection methods.

function. After these have been determined for each voxel, three important quantities can be calculated from the IRF [18].

$$CBF = Max(IRF) \tag{1.8}$$

$$CBV = \int_0^\infty IRF(t)\,dt \tag{1.9}$$

$$MTT = \frac{CBV}{CBF} \tag{1.10}$$

where CBF is the maximum value of the IRF. CBF, CBV and MTT satisfy the Central Volume Principle stated in Equation 1.4.

All these parameters are voxel based, which means their values are varying per voxel. After calculating the three parameters for all of the voxels of a brain, hemodynamic maps, such as 3D CBF, CBV and MTT maps (consisting of 2D slices), can be delivered.

There are two types of perfusion imaging commonly used in hospitals: CT perfusion imaging and MRI perfusion imaging.

1.6.1 CT Perfusion Imaging

Computed Tomography (CT) scanning, which is also called computerized axial tomography (CAT) scanning, is a medical procedure, involving rotating X-ray equipment and a digital computer, to obtain cross-sectional images of the body organs and tissues. CT scanning delivers slices of imaging, which can be used to form the 3D images (volume). CT scanning can deliver useful information such as whether there is a stroke, how serious the stroke is and which type of stroke it is.

The method, dynamic perfusion computed tomography (PCT or CT perfusion imaging), by which perfusion to an organ is measured using CT scanning, originated as early as 1980 by Leon Axel [19]. It is usually used for neuroimaging using dynamic sequential scanning of brain regions with an injection of a bolus of contrast material in order to evaluate how blood passes through a brain. After the images are acquired, deconvolution methods are used to process the raw source images and obtain quantitative hemodynamic information, such as CBF, CBV and MTT. Typically, a continuous scan of 40 to 45 seconds is performed, with a scan rate of one image (volume) per second. Regardless of the injection of contrast materials, the perfusion images are a sequence of regular non-perfusion images sampled as a time series with constant interval. In other words, perfusion image contains several volumes while non-perfusion one only contains one volume. Figure 1.3 is an example of CT perfusion image.

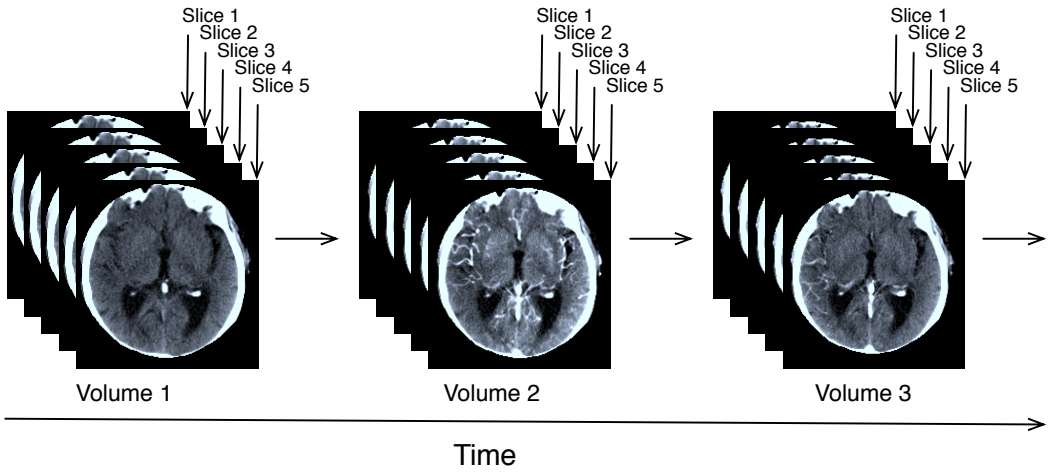


Figure 1.3: Perfusion Imaging Structure
This figure indicates the structure of a CT perfusion image.

In CT perfusion imaging, a delay is introduced by the relatively prolonged intravenous injection, this can be corrected for if the arteries are seen on the scan; it also helps to measure the background and noise level of the scan. After the delay, there will be a rise to a peak and then, the signal will drop off. Then, it may be followed by smaller peaks caused by noise or by recirculation.

Figure 1.4 shows the CT perfusion imaging workflow. The input of the workflow is CT perfusion source images. By collecting data from a given voxel at successive time points from the source images, a tissue time-concentration curve can be built for each voxel. The CT signal intensity, $S(t)$, during passage of bolus of contrast agent is related to the tissue concentration of the contrast agent, $C(t)$, according to the following equation:

$$S(t) = kC(t) + S_{baseline} + \varepsilon(t) \tag{1.11}$$

where k is an unknown constant of proportionality on the assumption that k is constant for all tissues and vessels. It divides out when the CBF is obtained as shown in Equation 1.3. $S_{baseline}$ is the background which is usually measured by the trimmed mean[4] of the beginning of the signal, ε is noise.

The main weakness of PCT is the higher radiation with CT perfusion imaging compared with standard un-enhanced CT methods. Since X-ray radiation increases the risk of cancer [20, 21], CT scanning is constrained by a tradeoff between image quality and the amount of radiation exposure to the patient. It also has a limited anatomic

[4]A trimmed mean or truncated mean is similar to the mean and median. It calculate the mean after discarding the high and low end of the given set.

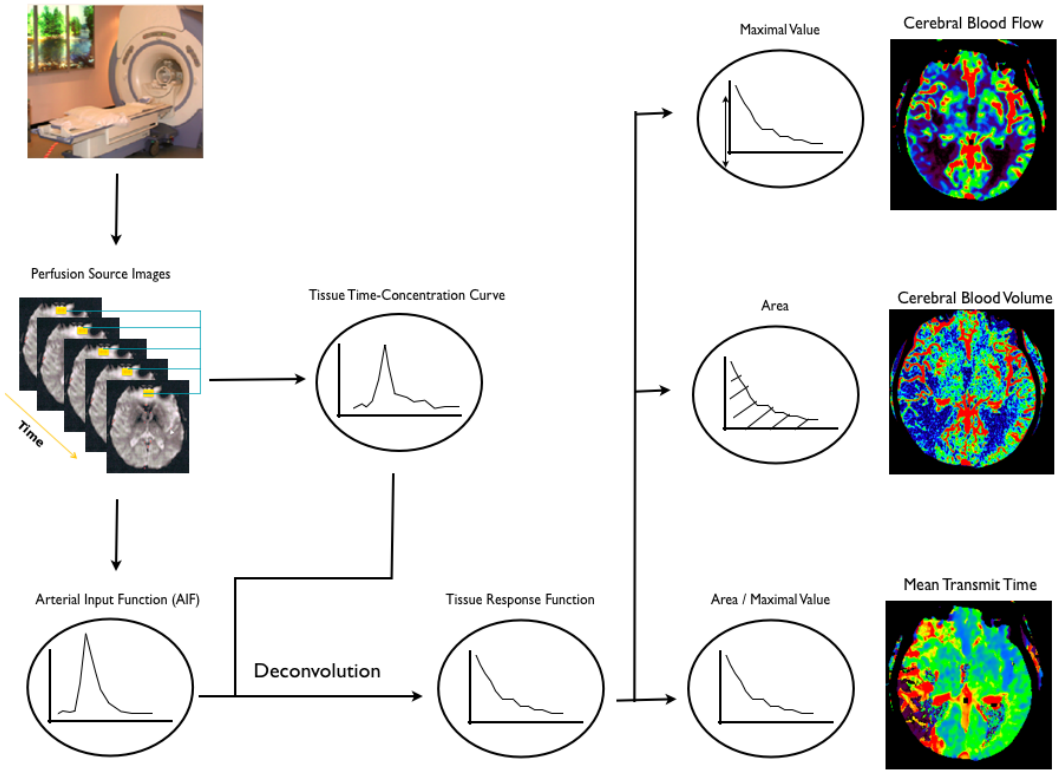


Figure 1.4: CT Perfusion Overview

This figure illustrates the workflow of CT perfusion imaging. There are four steps in the workflow: obtaining source images from a scanner, generating the tissue time-concentration curve for each voxels and AIF, calculating tissue response function using deconvolution and calculating hemodynamic maps.

coverage due to number of detector rings in currently installed devices in hospitals. However, compared with other brain imaging techniques such as PET, SPECT, XeCT, DSL, ASL and Doppler, perfusion computed tomography (CT) imaging remains one of the most easily accessible and widely available modalities for acute stroke patients [22, 23] and it enables the access to multiple perfusion hemodynamic parameters. PCT also has a good temporal resolution on the order of 1 second and a data acquisition duration typically of 40 seconds, which is shorter than other techniques due to the requirement to limit radiation explosure [24].

1.6.2 MRI Perfusion Imaging

The Magnetic Resonance Imaging (MRI) method is another medical imaging technique to visualize internal body structures in detail. The same as CT images, the initial images acquired from MRI scans are also multiple 2D slices which can be then be aligned and reformed into 3D volume.

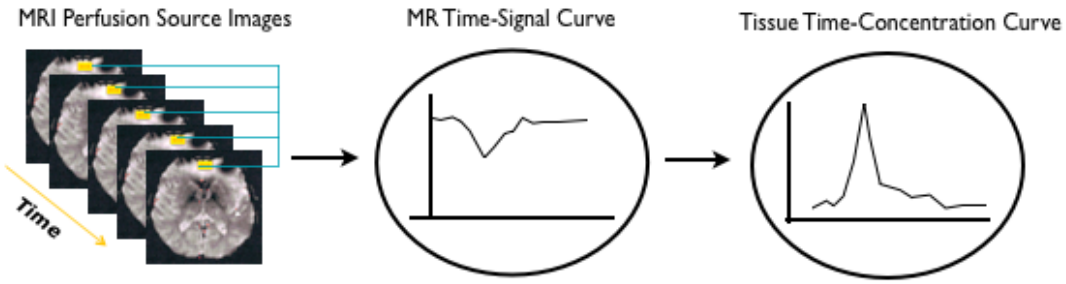


Figure 1.5: MRI Perfusion Imaging Signal Conversion

This figure demonstrates the extra step required by MRI perfusion imaging to convert MRI time-signal curves to tissue time-concentration curves.

MRI perfusion imaging is a type of MRI method. It is similar to CT perfusion imaging but acquires data using an MRI imaging scanner rather than a CT scanner.

MRI perfusion imaging has a similar analysis workflow to CT perfusion imaging. However, the procedure, which converts the measured function proportional to the instantaneous tissue time-concentration curve, is different as shown in Figure 1.5. Equation 1.11 for CT perfusion imaging does not apply in MRI perfusion imaging and the MRI signal intensity, $S(t)$, during passage of bolus of contrast agent is related to the tissue concentration of the contrast agent, $C(t)$, which satisfies the following equation:

$$S(t) = S_0 \exp(-kC(t)T_E) + \varepsilon(t) \tag{1.12}$$

where S_0 is the prebolus signal; k is an unknown constant. Similar to the CT case (Equation 1.11), we assume that k is constant for all tissues and vessels [25]; T_E is the echo time; ε is noise, which is assumed to be normally distributed. For each voxel, tissue time-concentration curve can be calculated from $C(t)$ as

$$C(t) = -\frac{\ln(\frac{S(t)}{S_0})}{kT_E} \tag{1.13}$$

Note that the noise, ε, is in $S(t)$, so the delivered $C(t)$ also contains the noise.

MRI scan is safe[5], because it uses magnetic and radio waves and patient is not exposed to X-ray or any other radiation. MRI perfusion imaging usually contains information for a longer scanning period (about 80 seconds), compared to the 30-40 seconds scanning period for CT perfusion imaging. Therefore, MRI perfusion imaging contains a longer start and tail than CT perfusion imaging. Furthermore, MRI perfusion imaging has a much better signal-to-noise ratio than CT perfusion imaging. MRI

[5]MRI is safe in general, but it is not safe for people with pacemakers for instance.

scans often differentiate normal and abnormal tissues clearer than CT scans. However, MRI imaging is not as easily accessible or widely available as CT imaging.

1.6.3 Delay of Contrast Agent Injection

With cerebral dynamic perfusion imaging using CT or MRI, the duration of the first passage of the bolus is approximately 5 to 20 seconds. The delay of the contrast agent injection provides an opportunity to determine the imaging baseline and noise level.

1.7 Perfusion Imaging Analysis Workflow

Ostergaard *et al.* [26, 27] and Wirestan *et al.* [28] have shown that accurate hemodynamic maps can be created using deconvolution of a tissue time-concentration curve and an AIF. From a CT or MRI scanner we get a series of brain images at different sampling times. For each voxel, we collect data at specific time intervals to build a tissue time-concentration curve of contrast agent intensity. This curve will be referred to as C and is separately determined for each voxel.

As stated before, the tissue impulse response function, IRF(t), which contains the information about brain hemodynamic quantities is the one we want to estimate for each voxel. The volume of fluid, C, C_a, and IRF satisfy the following convolution equation:

$$C = C_a \otimes IRF + \varepsilon = \int_0^t C_a(\gamma) IRF(t-\gamma) d\gamma + \varepsilon \tag{1.14}$$

where C_a is AIF vector, $\otimes$ denotes convolution and ε is the noise.

In this thesis, we used one popular method for forming the AIF matrix, which is created from the AIF vector follows [26]:

$$C_{AIF} = \Delta t \begin{pmatrix} C_a(t_1) & 0 & \cdots & 0 \\ C_a(t_2) & C_a(t_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_a(t_N) & C_a(t_{N-1}) & \cdots & C_a(t_1) \end{pmatrix} \tag{1.15}$$

where C_{AIF} is the AIF matrix, $(t_1, t_2, \cdots, t_N)$ are the sampling times, $(C_a(t_1), C_a(t_2), \cdots, C_a(t_N))$ is an arterial input function given as an input and Δt is the time scale. The convolution

in Equation 1.14 using Equation 1.15 can be formed as:

$$\begin{bmatrix} C(t_1) \\ C(t_2) \\ \vdots \\ C(t_N) \end{bmatrix} = \Delta t \begin{bmatrix} C_a(t_1) & 0 & \cdots & 0 \\ C_a(t_2) & C_a(t_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_a(t_N) & C_a(t_{N-1}) & \cdots & C_a(t_1) \end{bmatrix} \times \begin{bmatrix} IRF(t_1) \\ IRF(t_2) \\ \vdots \\ IRF(t_N) \end{bmatrix} \tag{1.16}$$

where $C(t_1), C(t_2), \cdots, C(t_N)$ are the intensity values measured from perfusion scan at different sampling time points and $IRF(t_1), IRF(t_2), \cdots, IRF(t_N)$ is the impulse response value for each time point. Again, this equation need to be applied to each voxel.

The effects of noise can be measured as follows:

$$||C_a \otimes IRF - C|| \tag{1.17}$$

where $||\bullet||$ indicates the vector norm. Specifically, the Euclidean norm is used here to measure the noise.

1.8 Deconvolution in Perfusion Imaging

Convolution is a mathematical operation that takes two functions as input and produces a new function that is typically viewed as a modified version of one of the input functions. The convolution process can be considered as an area overlapping of the two input functions. Another process, called deconvolution, is used to reverse the effects of convolution. The concept of deconvolution is widely used in signal processing and imaging processing, not only in perfusion imaging. Since one of the input functions (AIF) and the result of convolution (tissue time-concentration curve) can be obtained from perfusion imaging, deconvolution is required to calculate the tissue impulse response function, the other input function of convolution. There are different approaches to perform the deconvolution, such as Singular Value Decomposition, Gaussian process for deconvolution and orthogonal polynomials.

1.8.1 Truncated Singular Value Decomposition

Singular Value Decomposition (SVD) is one of the most popular techniques to solve deconvolution problems in perfusion imaging, since it allows the application of different discretization schemes and always gives real value[6]. Suppose C_{AIF} from Equation

[6]The 'real value' is in the mathematics context, indicates a non-complex value.

(1.15) is an m-by-n matrix[7], there exists a factorization such that:

$$C_{AIF} = U \cdot W \cdot V^T \tag{1.18}$$

where U is an $m \times m$ unitary matrix, W is $m \times n$ diagonal matrix and V^T is the transpose of an $n \times n$ unitary matrix V. A common convention is to order the diagonal matrix W in a decreasing order and this diagonal entries of W are known as the singular values of original matrix C_{AIF}. Elements on the diagonal matrix W are non-negative real numbers.

Then C_{AIF}^{-1} can be written as:

$$C_{AIF}^{-1} = V \cdot W^{-1} \cdot (U^T) \tag{1.19}$$

to solve the deconvolution problem in Equation (1.14), simply apply SVD:

$$IRF = V \cdot W^{-1} \cdot (U^T \cdot C) \tag{1.20}$$

However, as rows in C_{AIF} in Equation (1.14) are close to linear combinations, the deconvolution is an ill-posed problem. In other words, it is very sensitive to noise and thus similar inputs can lead to very different solutions.

Since there may be zero-value elements on the diagonal matrix W, its inverse matrix W^{-1} does not exist. The W^{-1} is calculated using following equation:

$$W^{-1}(i,i) = \begin{cases} \frac{1}{W(i,i)} & (W(i,i) > 0) \\ 0 & (W(i,i) = 0) \end{cases} \tag{1.21}$$

it may cause $WW^{-1} \neq I$ so W^{-1} is not the inverse matrix of W anymore. However, to keep it simple, the W^{-1} in Equation 1.21 is considered as the inverse matrix of W in the SVD.

Figure 1.6 is an example which illustrates the noise impaction in SVD. When calculating the inverse matrix of the diagonal matrices, comparing the W^{-1} (noise free inverse diagonal matrix) and $W^{'-1}$ (noisy inverse diagonal matrix), it is obvious that the noise is emphasised and dominates the inverse diagonal matrix. The huge deviations will finally disturb the measurement of IRF and hemodynamic quantities.

To minimize the noise impact, truncated SVD is introduced. In truncated SVD, a threshold is defined and elements of the diagonal matrix W with values smaller than this threshold will be set to zero in W^{-1} [26, 27], so Equation 1.21 in truncated SVD is written as:

$$W^{-1}(i,i) = \begin{cases} \frac{1}{W(i,i)} & (W(i,i) \geq Threshold) \\ 0 & (W(i,i) < Threshold) \end{cases} \tag{1.22}$$

$$W = \begin{pmatrix} 4 & & & & & \\ & 2 & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & 0 & \\ & & & & & 0 \end{pmatrix} \quad W^{-1} = \begin{pmatrix} 0.25 & & & & & \\ & 0.5 & & & & \\ & & 1 & & & \\ & & & 0 & & \\ & & & & 0 & \\ & & & & & 0 \end{pmatrix}$$

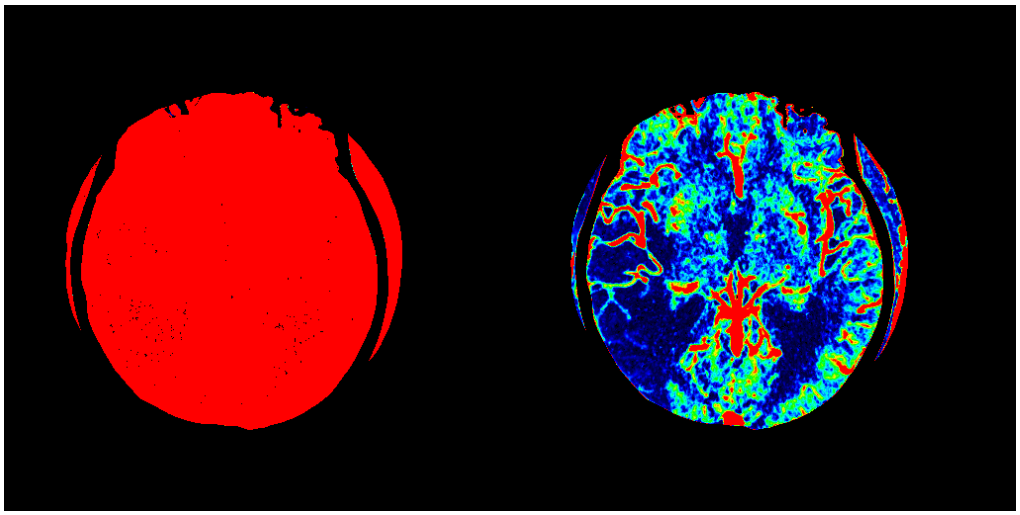
(a) Expected

$$W' = \begin{pmatrix} 4 & & & & & \\ & 2 & & & & \\ & & 1 & & & \\ & & & 0.2 & & \\ & & & & 0.1 & \\ & & & & & 0.05 \end{pmatrix} \quad W'^{-1} = \begin{pmatrix} 0.25 & & & & & \\ & 0.5 & & & & \\ & & 1 & & & \\ & & & 5 & & \\ & & & & 10 & \\ & & & & & 20 \end{pmatrix}$$

(b) Noisy

Figure 1.6: Noise Impaction in SVD

The equations on the top are the noise free diagonal matrix and its inverse matrix. The equations on the bottom are these matrices with noise.



(a) No Threshold (b) With Threshold

Figure 1.7: The Important of the Threshold

The figure on the left is a CBF map without using threshold and the figure on the right is a CBF with threshold equals to 0.15 of the maximal singular value. Both of the CBF maps are generated using the same data and threshold is the only difference. These two images are CT images. In this figure, values from low to high are mapped into color from black, blue, green, yellow and red.

Figure 1.7 is an example of how threshold improves hemodynamic quantities in the view of entire image. Without threshold, the CBF map shows nothing (Figure 1.7a) as all the information is overwhelmed by noise. With the given threshold, a much clearer

[7] In our case, C_{AIF} is a square matrix so that m equals n.

CBF map with details is delivered. The selection and validation of threshold will be described in Section 1.8.2.

1.8.2 Threshold Selection

There are two common ways to select the threshold for truncating the diagonal matrix in SVD.

The first one is to use a fixed threshold based on experience. This threshold is used for all of the voxels in a scan. A common choice is to set the fixed threshold to $0.15-0.2$ of the maximum singular value [28]. This threshold is widely used nowadays due to its low cost in terms of both of the processing time and implementation. In this thesis, truncated SVD with a fixed threshold of 0.15 of the maximum singular value is used in all of the following chapters.

The other way is to use the L-curve criterion (LCC), or generalized cross validation (GCV), to dynamically decide the threshold. Sourbron *et al.* in 2004 and 2007 [29, 30] stated that both LCC and GCV perform sufficiently well when applied with a variant of truncated SVD, which is known as standard form Tikhonov regularization [31]. LCC and GCV can reduce the oscillation in the solution and help in estimating a more accurate MTT value. They also declared that LCC and GCV are equivalently accurate, but GCV requires a lower calculation time than LCC. Compared with the fixed threshold method, dynamically selecting the threshold can improve the accuracy further, as it is self adjusted by exploiting noise level in different datasets. However, it is not as easy to be implement and the dynamic fitting process takes extra time (a few seconds) to execute. Although performance issue may be solved by parallel processing, they do not work at the low SNR - typical of CT perfusion images. Due to these reasons, we have not used dynamic threshold methods in the experiments.

1.8.3 Block-Circulant SVD

A block-circulant deconvolution method was proposed by Wu *et al.* in 2003 [32]. Block-circulant SVD is almost the same as SVD method mentioned in Equation 1.15. The only difference between these two methods is that, in this block-circulant SVD, the AIF matrix C in Equations 1.15 and 1.16 are replaced by a block-circulant matrix which satisfies the following condition:

$$C_{block\_circulant}(i,j) = \begin{cases} C_a(i,j) & (i \geq j) \\ C_a(N+i-j,0) & (i < j) \end{cases} \tag{1.23}$$

Equation 1.15 in the block-circulant SVD can be formed as:

$$\begin{bmatrix} C(t_1) \\ C(t_2) \\ \vdots \\ C(t_N) \end{bmatrix} = \Delta t \begin{bmatrix} C_a(t_1) & C_a(t_N) & \cdots & C_a(t_2) \\ C_a(t_2) & C_a(t_1) & \cdots & C_a(t_3) \\ \vdots & \vdots & \ddots & \vdots \\ C_a(t_N) & C_a(t_{N-1}) & \cdots & C_a(t_1) \end{bmatrix} \times \begin{bmatrix} R(t_1) \\ R(t_2) \\ \vdots \\ R(t_N) \end{bmatrix} \tag{1.24}$$

Block-circulant SVD is not sensitive to circulatory delay, which is often evident as a confounding factor when MR and CT data are acquired from patients with cerebrovascular steno-occlusive disease [32]. Therefore, the major advantage of block-circulant SVD is that it has a smaller tendency to underestimate flow when tissue tracer arrival is delayed relative to the AIF.

1.8.4 Gaussian Process for Deconvolution

Gaussian Process for Deconvolution (GPD), introduced by Andersen *et al.* in 2002 [33], is one of the other techniques to solve perfusion imaging deconvolution problems. The Gaussian process for deconvolution provides a reasonable IRF compared to SVD based methods. GPD can also estimate the noise level of data and automatically measure the uncertainty of the IRF estimation. GPD can provide a complete IRF which is much smoother than SVD. The major disadvantage of GPD is that, it contains a stronger regularization step than SVD, and this will affect the maximum IRF value more severely than SVD. Considering that the maximal IRF value will be used to determine the CBF value; it could be problematic if the deviation is too large.

1.9 Local AIF

Quantification of CBF using dynamic perfusion CT and dynamic susceptibility contrast MRI relies on the deconvolution of the tissue time-concentration curve and the AIF. Commonly, a single AIF, which is known as the global AIF, estimated from a major artery is used for each scan. However, the use of a global AIF can lead to a significant error since there maybe a delay of bolus between the selected major artery and the tissue of interest. Ostergaard *et al.* [26] stated that a delay of two seconds between assumed and real AIFs leads to an underestimation of flow at high flow values using the SVD deconvolution technique. In the experiment by Wirestam *et al.* in 2000 [28], they found out that the 0-1.5 seconds delay of AIFs leads to result with 2% difference in rCBF, which is significant.

Local AIF uses a different definition of AIF. There are multiple AIFs used in a single scan, each of them is determined by locally collecting data from the neighbors of interest voxel. It is introduced to minimize the error of the presence of bolus delay and dispersion between the artery and the tissue of interest [34, 35]. They declared that the use of local AIFs can avoid miscalculating the hemodynamic parameters due to the use of inappropriate AIF curves in deconvolution process.

The bolus of contrast material travels with relatively little dispersion while in the blood stream but disperse due to varying tissue type and condition in other tissue. The AIF curves, when using both local AIF and global AIF methods, are chosen from the voxels with the highest contrast ratio, i.e. the largest signal[8].

The determination of local AIF is similar to global AIF. They both use several criteria, such as peak value and arrival time, to calculate a score for each voxel and use this score to find out the artery in specific volumes. The difference is that global AIF technical using a single input (the whole brain) for all of entire volumes while local AIF technical applies different inputs (neighbor volumes near the voxel of interest) for different voxels.

1.10 Lesion Visibility

In perfusion imaging, the visibility of lesions is affected by two characteristics: inherent-contrast and size [8]. The inherent-contrast refers the fact to that the attenuation coefficient value of a lesion differs from that of the surrounding tissue. Generally, an easily visualized and high contrasted lesion has a much different attenuation value compared to the surrounding tissues. Size, the other characteristic of lesions, affects lesions in terms of details to be discerned. The smaller the object is, the greater the capabilities are required for detecting detail.

1.11 The Importance of Time

The phrase ‘time is brain’ was first introduced by Gomez in 1993 [36]. It emphasizes that human neurological system is rapidly and permanently damaged during stroke. Therefore treatment should be given to the patient as soon as possible.

[8] AIF is selected from voxels with the largest signal-to-noise ratios, but not necessarily from the one with the largest ratio.

Table 1.1: Brain Circuitry Loss during Stroke

	Neurons Lost	Synapses Lost	Myelinated Fibers Lost	Accelerated Ageing
Per Second	3.2×10^4	2.3×10^8	0.2 km	8.7 h
Per Minute	1.9×10^6	1.4×10^{10}	12 km	3.1 wk
Per Hour	1.2×10^8	8.3×10^{11}	714 km	3.6 y
Per Stroke	1.2×10^9	8.3×10^{12}	7140 km	36 y

This table shows the results of Saver's research [37]. It illustrates the damage produced during stroke in terms of the speed of brain circuitry loss.

Table 1.1 demonstrates quantitative measurement regarding how many brain cells, synapses and fibers are lost every minute during stroke as evaluated by Saver in 2006 [37]. Saver stated that, when treatment fails to occur, every 30 minutes, 57.6 million neurons die. In the same 30 minutes, your brain loses 41.4 billion synapses and 360 kilometres of axonal fibers. Furthermore, a brain loses as much neuron during each hour in which a stroke is untreated as it does in about 3.6 years of normal life.

As a result, a stroke is a medical emergency in which time is critical for a stroke patient; the sooner the treatment occurs, the less damage that will be caused to the patient's brain. So one of the objectives of stroke treatment is to open up the blocked vessels and restore the blood flow as soon as possible, before the damage has become severe. Treatments for different types of strokes are different. For example, the treatment appropriate for ischemic stroke exacerbates the damage caused by the hemorrhage. Diagnosis has to be made before treatment. This means that not only is the onset time critical, but also that the time required to deliver the results as well for diagnosis is critical.

1.12 Focus of Thesis

The scope of the study does not extend to a consideration of perfusion imaging acquisition and hemodynamic quantitative analysis. So, steps such as how images are acquired from the scanner, how they are reconstructed, registered and how to obtain hemodynamic quantities from the impulse response function, are all performed using traditional methods which we did not try to modify or optimize.

In this thesis, we mainly focus on the process that starts from perfusion source images and ends when the hemodynamic maps are generated.

1.13 Synopsis of Thesis

The propose of my PhD research is to develop novel methodologies for improving the efficiency and quality of brain perfusion-imaging analysis so that clinical decision can be made more accurately and in shorter time. In other words, my PhD research involves the use of computer science methodologies to solve medical imaging problems.

This thesis that we will support with experimental evidence is that the use of parallel algorithms, general purpose graphics processing units and noise reduction methods will significantly improve the quality of hemodynamic maps and deliver this diagnostic information substantially more rapidly than currently employed methods. We have also investigated an automatic lesion area detection method, which generates diagnosis result without expert's involvement. Results derived from our methods are compared with results from traditional methods. Criteria, such as processing time, whole brain summary statistics, hemodynamic maps and experts' opinion, are used to evaluate the results.

1.14 Outline of Thesis

The remainder of the thesis is organized as follows:

Chapter 2 describes our GPGPU based parallelized approach to optimise the performance regarding perfusion-imaging analysis using local AIFs. The performance of our GPGPU implementation is compared with the original serial implementation and parallel implementations based on conventional CPUs. Part of this chapter was published in [38, 39].

Chapter 3 elaborates a temporal-information-based noise reduction method using Gaussian-process regression and its variant, multiple-observation-Gaussian-process regression, which is a combination of spatial filters and temporal filters. Results derived from our methods are compared with results of medical-image specific noise reducing filters. Part of this chapter is based on the work published in [40].

Chapter 4 presents an automatic tissue segmentation method that classifies the tissue in each voxel. This approach uses correlation coefficient tests to validate tissue time-concentration curves for each voxel. Thus its segments voxels into potentially healthy, dead and penumbra voxels.

Chapter 5 summarises the objectives of this thesis, presents its main findings and discusses future work.

Chapter 2

Performance Speed Up Using GPGPU

[Parts of this chapter have been published as 'A Parallel Deconvolution Algorithm in Perfusion Imaging' in Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 [38] and as 'Parallel Perfusion Imaging Processing Using GPGPU' in Computer Methods and Programs in Biomedicine (CMPB), 2012 [39].]

2.1 Introduction

This chapter presents a parallel implementation of brain perfusion-imaging analysis using GPGPU. In such analysis, there are thousands of voxels to be deconvolved what constitutes a heavy computational task that can be tackled by using parallelism. The idea is to separate the computational tasks into voxel-based small tasks, distribute them to different GPU threads and obtain performance improvements from data parallelism. In order to evaluate the performance improvement, we also implemented a serial version and other CPU-based parallel version. This chapter will focus on the performance issue, thus results delivered by serial and parallel implementations are identical and the running time will be the only criterion when judging the quality of different implementations.

2.1.1 Motivation

The artery input function is one of the two input functions of the deconvolution, as stated in Sections 1.6.1 and 1.6.2. In clinical practice, the most commonly used AIF selection technique is global AIF, which determines a single AIF from voxels near a major artery feeding the brain for the entire brain. However, the global AIF technique

is based on the assumption that the contrast agent reaches every voxel of the brain at the same time and that the bolus preserve its shape; and even in the case of healthy brain, there is delay and dispersion which means this assumption is incorrect; a stroke will further increase this type of error. As a result, using a global AIF for the entire brain is not very accurate [41, 42].

The other AIF selection technique in use is local AIF [35, 43, 44, 34]. In the local AIF technique, different AIFs are used for a single scan, compared to the single AIF for the whole brain in global AIF technique. Each local AIF is generated by measuring a small set of blood vessels in a specified area near the voxel of interest. Lorenz *et al.* [35] have shown that localized AIFs are feasible and provide more useful perfusion results.

In this chapter, we will not discuss the advantages and disadvantages of global AIF and local AIF techniques (because different experts have different preferences). The focus of this chapter is on the performance (in terms of the processing time) of the perfusion-imaging analysis. Since the AIF matrices (Equation 1.15, Equation 1.23, etc.) for different voxels in a single scan are identical using global AIF technique, the operations of decomposing AIF matrices and related operations can be reused. As a result, only one decomposition is required for the entire scan. However, the use of local AIFs involves the use of different AIF matrices, no matter what kind of deconvolution techniques being used, and incurs in a large number of matrix decomposition operations accordingly. As a result, using local AIFs leads to fairly slow performance. In the worst case, the perfusion-imaging analysis takes more than half an hour compared with the running time of global AIFs based methods which is a couple of minutes.

According to Saver's experiment in 2006 [37] (Section 1.11), during 30 minutes, 57.6 million neurons die. In the same minutes, your brain loses 41.4 billion synapses and 360 kilometres of axonal fibers. Since a stroke is a medical emergency and every second counts, the sooner results are delivered in diagnosis, the less damage will be caused to a patient's brain. Obviously, half an hour is not a reasonable lapse of time for clinical diagnosis.

Furthermore, since the image resolution keeps increasing in perfusion imaging, the demand for performance speedup is growing.

A parallel implementation of perfusion-imaging analysis, which brings performance speedup without quality loss, is a potential solution to increase the usability of the local AIF technique in perfusion imaging. To prevent losing image quality, the parallel implementation should deliver identical results to the serial implementation.

In addition to parallelization, performance optimizations such as data reconstruction, which will be mentioned in Section 2.2.2, are also used in both of our serial and parallel implementations. In this chapter, we will present such a parallel implementation of perfusion-imaging analysis.

2.1.2 Parallel Computing

Parallel computing is a state-of-the-art technique to handle computational tasks. Traditionally, programs are written for serial computing on one central processing unit (CPU), which means programs are constructed and implemented as a serial stream of instructions, instructions are executed one after another and only one instruction can be executed at a time. In parallel computing hardware with multiple processing units, either multiple CPUs or a multiple-core CPU is used. Programs are split into discrete small sub tasks that can be solved concurrently. Within each sub task, a program is still formed from a series of instructions, just as in a serial computing program. In other words, the essence of a parallel task is a set of serial tasks. Between different parallel sub tasks, different parts can be executed simultaneously on different processors (or cores). Furthermore, multiple instruction, multiple data (MIMD) is the most popular technique used to achieve parallelism. Using MIMD, processors work asynchronously and independently, which means different processors can execute different instructions on different data.

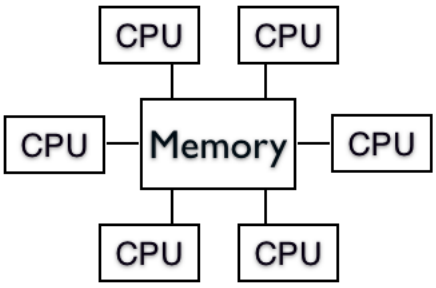
Traditionally, parallel computing has been employed mainly in high performance computing. However, with the spread of multi-core CPUs, parallel computing has become available for all of the application domains and computational contexts.

2.1.3 Parallel Architectures

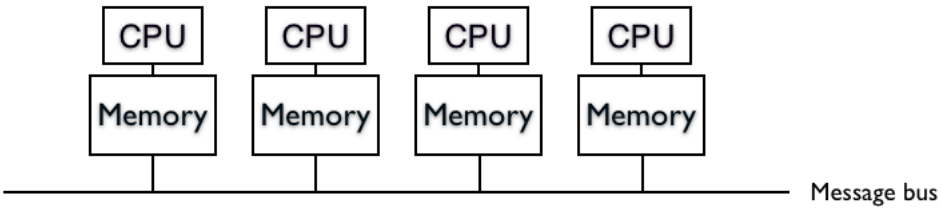
According to the communication methods between different parallel threads, parallel methods can be roughly classified into three categories: shared-memory parallel architecture, distributed-memory parallel architecture and hybrid-memory parallel architecture.

2.1.3.1 Shared-Memory Parallel Architecture

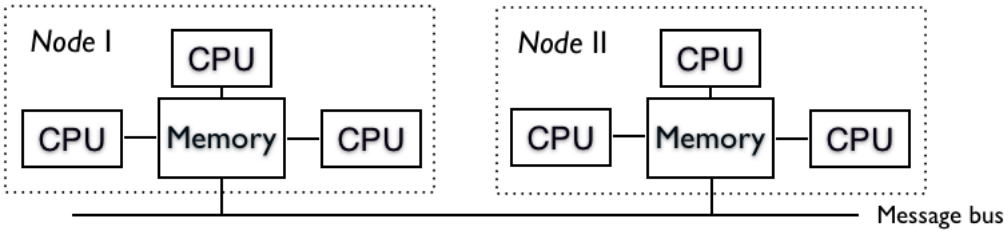
Shared-memory parallelization is a type of parallel architecture based on shared-memory architecture (Figure 2.1a). Shared-memory architecture refers to systems where all of



(a) Shared Memory



(b) Distributed Memory



(c) Hybrid Memory

Figure 2.1: Parallel Computer Memory Architectures

Table 2.1: Parallel Architectures Comparison

	Shared Memory	Distributed Memory	Hybrid Memory
Scalability	Poor	Good	Good
Programmer Friendly	Good	Poor	Medium
Cache Coherency	Poor	Medium	Medium
Communication Overhead & Speed	Good	Poor	Good within nodes Poor between nodes

the processors can access all of the memory as a global address space. Different processors operate independently but share the same memory resources. Shared-memory architecture is commonly implemented on symmetric-multiprocessor (SMP) machines. Processors in an SMP machine are identical and each processor has the same priority

and same access times to memory. If one processor updates a cache location in the shared memory, the hardware will inform all of the other processors about the update.

The advantages of shared-memory parallel architectures are that the cost of sharing data is low and it is programmer friendly. The major disadvantage of this architecture is the lack of scalability between memory and CPUs. Increasing the number of CPUs is limited by the CPU bandwidth (the bandwidth between the memory and the CPU) and extra overhead for cache-coherent systems. Hence, it becomes increasingly difficult and expensive to design and produce shared-memory machines with ever growing numbers of processors [45].

One of the most popular shared-memory programming languages is OpenMP (Open Multi-Processing) which was first released in October 1997 by the OpenMP Architecture Review Board [46, 47]. It supports multi-platform, shared-memory multiprocessing programming in C, C++ and Fortran on most processor architectures and operating systems, including Linux, Unix, AIX, Solaris, Mac OS X, and Microsoft Windows. General-purpose computing on graphics processing units (Section 2.1.4) also belongs to this architecture.

2.1.3.2 Distributed-Memory Parallel Architecture

Message-passing parallelization uses the distributed-memory architecture (Figure 2.1b) in which a set of tasks use their own local memory during computation. The communication between tasks is achieved by sending and receiving messages. Both sender and receiver tasks are required to participate during the communication.

The advantages of a distributed-memory parallel architecture are that both memory and the number of processors are scalable so it can be expanded with ease. Besides, each processor has its own memory and cache so the memory access is rapid without any overhead from cache coherency. The main disadvantage of this architecture is that a programmer is usually responsible for lots of the details regarding data communication between processors, which leads to extra and complicated programming compared with the shared-memory architecture. Besides, the data communication is slower so it requires the programmer to be very careful when partitioning tasks into subtasks.

MPI (Message Passing Interface), which was first released in June 1994 [48, 49], is a language-independent communications protocol used to program message-passing parallel computers. It has become very popular for writing distributed-memory algorithm.

2.1.3.3 Hybrid-Memory Parallel Architecture

The largest and fastest super computers in the world today are using a hybrid-memory architecture. There are different hybrid-programming methods using the hybrid-memory architecture (Figure 2.1c). One very popular one is *hybrid masteronly* [50] in which outer loops are parallelized using MPI and inner loops are parallelized with OpenMP. In this method, a computer can be divided into sub-machines. Within each sub-machine, it uses a shared-memory architecture and works as an SMP node, so that processors share the memory within this sub-machine. Processors in a sub-machine know only about their own memory but not the memory on another sub -machine. It uses message-passing architecture between different sub-machines, so communications between different sub-machines require message passing.

The advantages of hybrid memory architecture is that it is well suited to the trend of current hardware environment with multi-core or many-core machines clustered together. It can provide all of the advantages brought by shared-memory architectures within its SMP nodes and also benefit from message-passing architecture, so it has a good scalability. However, its main problems are that all of the other threads are sleeping while master threads communicate which leads to an expensive communication cost; it also requires the programmer to make a lot of extra effort to manage these communications. A summary of the advantages and disadvantages of the three memory architectures can be found in Table 2.1.

The hybrid-memory architecture can only be exploited successfully for problems which can be partitioned into subtasks which effectively fill sub-machines. This is a rather specialist set of computational problems. The perfusion imaging analysis does not fall into this category of problems. Therefore, implementation based on hybrid architecture is not used in this chapter.

2.1.4 General Purpose Computing on Graphics Processing Units

A graphics processing unit or GPU (which is also called visual processing unit or VPU) is a specialized circuit designed to rapidly operate memory in order to accelerate the building of images in a frame buffer intended for output to a display. The term GPU was introduced in August 31, 1999 by NVIDIA. NVIDIA introduced its GeForce 256 as “the world’s first ‘GPU’, or Graphics Processing Unit, a single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that was capable of processing a minimum of 10 million polygons per second” [51]. The use of

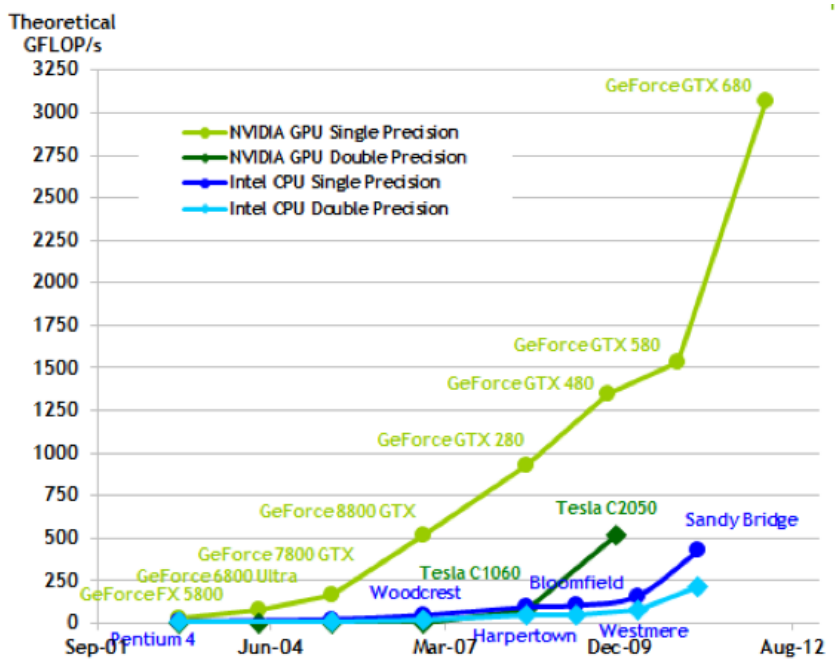


Figure 2.2: GPUs vs. CPUs

This figure [53] indicates the trend of FLOPS performance in both GPUs and CPUs.

synthesised dynamic images that force a frequent re-computation of graphics produce a heavier demand for graphics computation power than ever known before. This applies in video games, as the realism of the scene has to be achieved while keeping pace with the rate of state changes in the game that need to be perceived by human player in order that they can participate actively in the game. With the advent of the GPU, these computing tasks can be offloaded from the CPU onto the GPU. Todays market leaders for GPUs are AMD (under the ATi label), Intel and NVIDIA. According to a retail study by Current Analysis [52] in September 2006, more than 90% of new desktops and laptops had integrated GPUs.[1]

In the programming guide published by NVIDIA in 2012 [53], a definite trend is clear that GPUs have evolved into highly parallel, multithreaded, many core processor with tremendous computational power. In Figure 2.2, it can be observed that the growth of GPU processing power still follows Moore's Law. The increase of floating-point operations per second (FLOPS) of a GPU is approximately doubled every two years. Gordon E. Moore stated that the CPU clock speed is close to its theoretical upper bounds, it is no longer possible to achieve such performance improvement by increasing the speed of a single CPU [54]. Hence, a GPU becomes more and more

[1] An integrated GPU works the same as a dedicated graphics card but is less powerful.

suitable for large scale computational tasks.

GPUs originated as dedicated graphics generation co-processors. However, in the early 2000s, the idea that using GPUs as a more general processor capable of executing scientific computations was introduced. General-purpose computing on graphics processing units (GPGPU) [55] are state-of-the-art approaches to many computing applications. GPGPU can be regarded as using GPUs as multiple-core CPUs, where each core has a weak computational capability but the number of cores is large (thousands or more). Unlike using general CPU applications, which only need the involvement of one or more CPUs, a CPU is also required as well as GPUs when using GPGPU. Tasks such as input and output of data, data transfer between CPUs and GPUs and GPU thread scheduling are all under the control of a CPU thread. The CPU thread can be thought of as the master thread and the GPU threads as the worker threads.

GPGPU provides a highly parallel computing environment due to their huge number of computing cores and constitute an affordable, high-performance computing platform. More specifically, GPGPU is especially well suited to address data parallel computation problems, whose task is executed on a large number of data elements in parallel; especially for those tasks which can be split into single-instruction, multiple-data (SIMD) subtasks. For example, GPGPU is good at handling matrix operations since the same transformation is executed on every element within the matrix and there is almost no dependence between different elements. It is important for GPGPU applications to have high arithmetic intensity[2] in order to keep the memory access latency low [56].

Both the CPU and GPU architectures have their advantages. The CPU architecture has been designed for serial tasks while the GPU architecture is ideal for parallel tasks. In the aspect of clock speed, the clock speed of the best GPU is only 1.6GHz (NVIDIA GTX 580 released November 9, 2011), while a CPU can double that clock speed with ease. In the world of GPUs versus CPUs computing, many studies claim that GPGPUs deliver speedups between 10 and 1000 over multicore CPUs [57, 58, 59, 60]; however, Intel recently claimed that the performance gap between GPUs and CPUs narrows to only 2.5$\times$ on average [61]. Regardless of the difference between the two views, GPUs do provide a performance improvement.

Since GPU shows its advantages in computational capability and its ability to improve performance has been proved, it is worth exploring GPGPU in perfusion imaging analysis processing.

[2] Arithmetic intensity is defined as the number of operations performed per bit of memory transferred.

2.1.5 Programming Languages for GPGPU

At the dawn of GPGPUs, parallel algorithms on GPUs were implemented using special-purpose OpenGL-based or DirectX-based techniques, which required programmers to learn graphics programming first.

Compute Unified Device Architecture (CUDA) is a parallel computing architecture developed by NVIDIA in 2006 [62] with an associated software toolkit to solve the above awkward situation. It is the entry point for developers who prefer high-level computer programming, compared with Open Computing Language (OpenCL), which is the entry point for developers who want low-level Application Programming Interfaces (APIs). The CUDA programming architecture is very well suited to expose the parallel capabilities of GPUs.

C for CUDA offers programmers a simple way to write C-like programs for GPGPUs. It consists of a set of extensions to the C language for code running on CPUs and a runtime library for code running on GPUs. It significantly reduces the runtime overhead of GPGPU applications. As a result, CUDA has become one of the most popular programming languages for GPGPU programming. In addition, although it access GPUs via high-level APIs compared with other architecture such as OpenCL, it also allows programmers to use low-level APIs to avoid the overheads common with graphics APIs. CUDA code is hardware independent[3]. Its code does not need too much expert attention to be moved to different hardware, but it may require experts to adjust its configuration based on hardware to achieve the best performance. The level of abstraction and quality of semantic definition of the code and APIs for CUDA is sufficient to achieve code mobility.

2.1.5.1 CUDA Data and Control Flow

Compared with other CPU-based parallelization techniques, CUDA is very different in terms of its memory architecture. In CUDA, there are two types of memory: CPU memory and GPU memory; two types of function: functions that run on a CPU and functions that run on a GPU. The different types of memory cannot see each other; to put it differently, a CPU function can only access the CPU memory and a GPU function can only access the GPU memory. Thus, input data for a GPU function has to be copied from CPU memory to GPU memory before the GPU function can be executed and the

[3] CUDA was designed for NVIDEA hardware, but the latest CUDA allows CUDA code to be ported to non-NVIDEA hardwares.

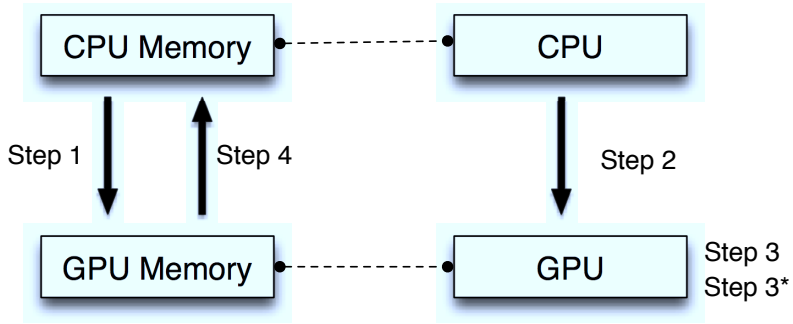


Figure 2.3: CUDA Data and Control Flow

This figure indicates the four steps of a CUDA data and control flow.

results of a GPU function also need to be copied back from GPU memory to CPU memory after the GPU function has finished via specific explicit CUDA functions. These data transferring functions are managed by the CPU thread (the 'master' thread).

Figure 2.3 is a typical example of GPGPU execution workflow in CUDA:

1. A CPU thread copies data from main memory to GPU memory.
2. A CPU thread instructs GPU threads to start processing.
3. GPU threads execute in parallel on different GPU cores. (Only GPU threads that are assigned tasks are running, the remaining threads will go directly into waiting in Step 3*)

3.* The CPU thread and all of the idle GPU threads wait for completion of the running GPU threads. This step happens at the same time as step 3.

4. The CPU thread copies the results from GPU memory to main memory.
5. The CPU thread acts on the results, and may return to step 1 in order to execute another GPU function.

2.1.5.2 CUDA GPU Threads

CUDA GPU threads execute independently. These threads must be able to be executed in any order, in parallel or in series. In a CUDA program, each of the threads is given a unique thread ID to identify itself through a **threadIdx** variable. The **threadIdx** is a three component vector, so that threads can be identified by using either a one, two or three dimensional index. Thus data structures such as vector (1D), matrix (2D) and 3D array can be easily managed under the one, two or three dimensional thread blocks, respectively. Usually, all of the threads within a block are assigned to the same processor core. Due to the limited memory resources of the GPU processor core, the

Table 2.2: Definitions in Pseudo Code

Input	4D MRI or CT image data stored in a NIfTI-format file [63].
Output	A set of CBF, CBV and MTT colored maps.
Time	the number of time intervals.
Dim1, Dim2, Dim3	the size of each dimension.
Size	the size of each 3D brain image which equals $Dim1 \times Dim2 \times Dim3$.
A()	a 4D array used to store data directly read from brain images.
A'()	a 4D array used to store data after reorganization (and denoising).
IRF	a 1D array used to temporarily store the result of deconvolution.
CBF(), CBV(), MTT()	3D arrays used to store the analyzed result.
CPU.A	Parameter A is stored on CPU memory.
GPU.A	Parameter A is stored on GPU memory.
$GPU.A \leftarrow CPU.A$	Copy data from CPU memory to GPU memory.
$CPU.A \leftarrow GPU.A$	Copy data from GPU memory to CPU memory.
$GPU\ A \leftarrow B$	Operation $A \leftarrow B$ is executed on the GPU.

number of threads per block is limited by the memory of processor cores. In CUDA, there is a hard limit on the number of threads per block. In the latest version (CUDA 4.0), this limit is of 1024 threads, while in CUDA 3.0 it was 512. Furthermore, tasks are expected to be divided into equally shaped thread blocks[4], so that the total number of threads is equal to the product of the number of threads per block and the number of blocks.

GPU threads within the same thread block can cooperate with each other simply by using shared-memory and synchronize their execution to coordinate memory accesses[5]. At the level of the entire task, all of the GPU threads can be synchronized by creating a barrier in the CPU thread (the master thread). Any thread must stop at the barrier and cannot proceed until all of the other threads reach this barrier.

2.2 Perfusion Imaging Algorithms Analysis

The perfusion imaging analysis comprises five steps: source image loading, data reorganization, denoising (optional), deconvolution and generation of results. This section determines whether these steps are suitable for parallelization as well as which parallel method is appropriate by analyzing their features.

[4]Equally shaped thread block means each block has the same number of threads.

[5]It is the users' responsibility to synchronize the shared memory. In our case, there is no need to synchronize the shared memory as different threads operate on different memory.

ALGORITHM 1 - SERIAL PERFUSION IMAGING ANALYSIS

```
1  A(1 : Time, 1 : Size) ← 4D MRI or CT image data
2  if (DoImageDenoising)
3     then A'(1 : Size, 1 : Time) ← denoiseAndReorganize A(1 : Time, 1 : Size)
4     else A'(1 : Size, 1 : Time) ← reorganize A(1 : Time, 1 : Size)
5
6  for i ← 1 to dim1 × dim2 × dim3
7        do {
8           Generate localAIF(i, 1 : Time)
9           IRF(1 : Time) ← Deconvolution result (A'(i,1:Time), localAIF(i, 1:Time))
10          CBF(i) ← Max(IRF(1 : Time))
11          CBV(i) ← Sum(IRF(1 : Time))
12          MTT(i) ← CBV(i)/CBF(i)
13          }
14 CBF colored map ← CBF(1:Size)
15 CBV colored map ← CBV(1:Size)
16 MTT colored map ← MTT(1:Size)
```

2.2.1 Definitions in Pseudo Code

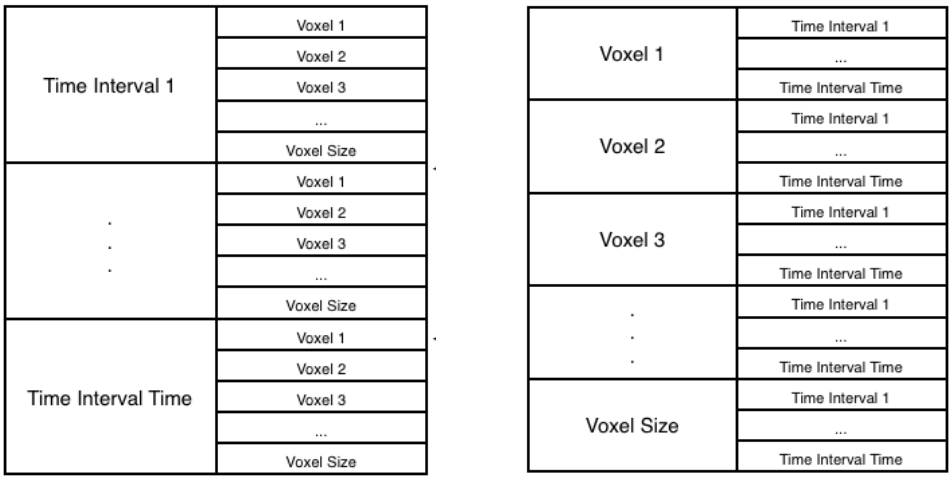
Table 2.2 contains the definitions of the variables and operations in the pseudo code.

2.2.2 Serial Perfusion Imaging Analysis

The algorithm for perfusion-imaging analysis without parallelization can then be written as Algorithm 1.

Source image loading The first step (Line 1) is to load the MRI or CT imaging data stored in neuroimaging informatics technology initiative (NIfTI) format file [63]. The computational complexity of step one is $O(Dim1 \times Dim2 \times Dim3 \times Time)$.

Data reorganization For each voxel, to generate tissue time-concentration curves in the deconvoluting step (Line 4) requires data from all of the time intervals. However, images are originally stored in a different way with the voxels grouped by time interval (Figure 2.4a). This kind of data structure will dramatically increase cache swap overhead. So the second step is to reorganize data from the form of $[time][Dim3][Dim2][Dim1]$ into the form of $[Dim3][Dim2][Dim1][time]$ (Figure 2.4b) to maximise data localization. The computational complexity of this step is $O(Dim1 \times Dim2 \times Dim3 \times Time)$.



(a) Original (b) Reorganized

Figure 2.4: Data Structure

The figure on the left shows the way data structured in the source file. The figure on the right represents the expected data structure which achieves the maximal localization.

Denoising (optional) As blood always flows from one cell to its neighbours, the intensity values should be continuous. This allow us to use an image-level denoising method (Line 3) such as applying 2D and 3D weighted mean filters. The computational complexity of this step is also $O(Dim1 \times Dim2 \times Dim3 \times Time)$. In the implementation, the reorganization and denoising steps are combined together as a *Denoising & Reorganization* step. Since denoising is not the main concern of this chapter, this chapter only uses simply denoising filters. More noise reduction issues will be presented in Chapter 3.

Deconvolution Lines 6 to 13 perform and use the deconvolution. This operation is executed voxel by voxel. Local AIFs, one of the input for deconvolution, are determined using the method mentioned in Section 1.9 (line 8). The most expensive part in the deconvolution is to decompose local AIF matrices using truncated singular-value decomposition. Since the AIF matrices formed using Equation 1.15 have a size of $time^2$, the computational complexity of decompositions is $O(time^3)$ according to our implementation and this is confirmed by J. Tesic *et al.* [64]. The computational complexity of deconvolution can be roughly considered as the same as the one for decomposition: $O(Time^3)$.

Furthermore, as voxel-based deconvolution needs to be repeated $Dim1 \times Dim2 \times Dim3$ times, the overall computational complexity is $O(Dim1 \times Dim2 \times Dim3 \times Time^3)$.

This is the most expensive part of the whole workflow, more details can be found in Section 2.3.2.

Generation of results The last step (Lines 14 to 16) is to produce parametric maps using the results generated from deconvolution. The computational complexity of this step is $O(Dim1 \times Dim2 \times Dim3)$.

Overall The steps *source imaging loading, data reorganization, denoising* and *Generation of Results* can be assumed to be small compared to the *deconvolution* step provided that $time > 2$. This assumption is always true in perfusion imaging where time is on the order of $10^1 - 10^2$. Hence, the overall computational complexity for perfusion-imaging analysis is $O(2 \times Dim1 \times Dim2 \times Dim3 \times Time + Dim1 \times Dim2 \times Dim3 \times Time^3)$ which can be considered as $O(Dim1 \times Dim2 \times Dim3 \times Time^3)$ which is the same as the computational complexity of deconvolution step. The serial algorithm is designed and implemented based on our understanding of current exist deconvolution and decomposition algorithms. It is possible that there exist better deconvolution algorithm, but our algorithms provides similar performance to the experiments reported by Lorenz *et al.*[35]. The running time for the serial algorithm will be reported in section 2.3.3.

2.2.3 Parallelization Feasibility Analysis

In this part, the feasibility of parallel perfusion-imaging analysis will be presented.

2.2.3.1 Un-parallelized Parts

The source image loading and generation of results steps consist mainly in reading and writing files. Hence, it is important to evaluate the size of input and output files in order to predict their running time.

For a large input dataset which has 128×128 voxels per slice, 22 slices per time interval and 80 time intervals, in total, there will be about 2.9×10^7 input elements and 3.6×10^5 output elements[6]. Each input element is a *short* type variable[7] which requires 2 bytes memory. There is a header file for each time interval which describes image dimensions, voxel dimensions, voxel data type, image orientation, etc. The header

[6] Assuming that the output files still have a resolution of 128×128 (no spatial filter is used), so that there is one output element per voxel

[7] Sometimes the input element is a *double* type variable which takes 8 bytes each.

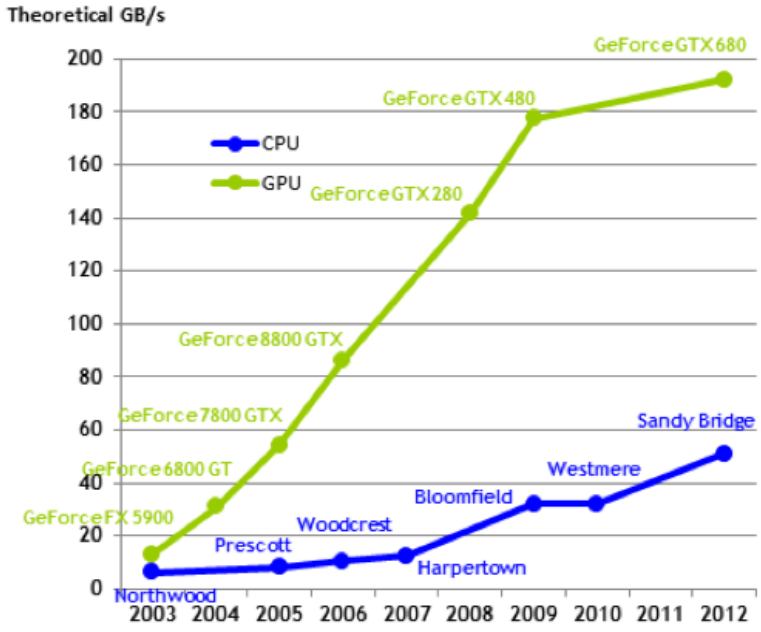


Figure 2.5: CPU bandwidth

This figure (From CUDA Programming guide [53]) indicates the trend of CPU bandwidth.

sizes of both input and output images are relatively small, compared with the size of the whole image. Hence, the size of input and output images can be consider as roughly 55 megabytes for input images and 1 megabyte[8] for each output hemodynamic quantity. The observed sizes of input and output files in our experiments are in agreement with the aforementioned expected values.

Figure 2.5 shows the trend of CPU bandwidth in recent years; it shows that a CPU can achieve a bandwidth $\geq$ 20 GB per second with ease. As the size of both input and output files are comparatively small compared to the bandwidth of a CPU, theoretically, it should only takes a few milliseconds in these two steps, which is already fast compared with other steps in the hemodynamic computation. At present, the two steps consist of I/O operations whose parallelized version is not supported by CUDA.[9] The running times of these memory operations can be found in Section 2.3.2.

2.2.3.2 Parallelized Parts

There are several options when separating perfusion-imaging analysis into parallel tasks. For example, if a task is to decompose one hundred unrelated matrices, it can be

[8]The hemodynamic maps are delivered in bitmap images. Each element takes 3 bytes.

[9]In fact, the data loading and writing rates are often limited by the performance of the storage system. Disk I/O bandwidth can also limit the performance but considering that the sizes of input and output files are relatively small, reading and writing operations are still cheap.

Table 2.3: Parallelism Method for each Step

Step	Parallelism Method
Source image loading	No parallelism
GPU memory copy in	No parallelism
Data reconstruction	Lower-level parallelism
Denoising	Lower-level parallelism
Deconvolution	Upper-level parallelism
GPU memory copy out	No parallelism
Generation of results	No parallelism

The first column of this table indicates the name of each step. The column on the right describes the parallelism method used for each step.

parallelized in three different ways. The first option is to apply parallelism inside each decomposition, each sub-decomposition is in charge of operations corresponding to a part of a matrix and different matrices are decomposed one after the other. This will be referred to as 'lower-level parallelism'. The second option is to separate the task into one hundred subtasks, where each subtask contains the decomposition for one matrix, and all of the one hundred different decompositions are performed concurrently in one hundred different threads. This will be called 'upper-level parallelism'. The third option is a combination of the first two methods. This splits each decomposition into ten parts and then uses 1,000 threads to execute all of the decompositions in parallel. This will be called 'hybrid parallelism'.

Table 2.3 shows which parallelism method is applied to each step in perfusion-imaging analysis for the reasons stated below.

Lower-Level Parallelism

Data reorganization After the images are loaded into memory, they are stored in a four dimensional array which contains millions of elements. Data reorganization can be considered as the transformation of an extremely large four dimensional matrix. As CUDA delivers good performance for matrix operations, it is expected that its lower-level parallelism will be the optimum choice.

Denoising As this chapter focuses on performance rather than accuracy, only simple spatial-filter-based denoising (weighted mean filter) is taken into consideration. As for data reorganization, the denoising can be considered as a transformation of a

Table 2.4: Computation time for SVD (in seconds)

Matrix Size	MATLAB	MKL	GPGPU
64 x 64	0.01	0.003	0.054
128 x 128	0.03	0.014	0.077
256 x 256	0.21	0.082	0.265
1K x 1K	72.0	11.255	3.725
2K x 2K	758.6	114.625	19.6
4K x 4K	6780.0	898.230	133.7

The column on the left indicates the size of each matrix; the second column is the time to decompose the matrix using MATLAB and third column is the result for Intel math kernel library LAPACK [65]. The fourth column shows the results of Lahabar's work that using a GPGPU to perform the decomposition.

large four dimensional matrix. Therefore, it can also exploit lower-level parallelism.

Furthermore, the data reorganization and the denoising parts can be combined into a single complex matrix transformation. This transformation then uses lower-level parallelism. So this step is actually a combination of data reorganization and denoising, which is referred to as *Denoising & Reorganization* step or *Denoising* step in this chapter.

Deconvolution At the lower level of the perfusion-imaging analysis, it is possible to use a GPGPU inside each deconvolution. Looking into the deconvolution for one voxel, the dominant part of the deconvolution is a matrix decomposition, which is followed by some matrix multiplications. As the decomposition dominates the running time, to simplify, a deconvolution can be considered as a decomposition.

GPGPUs can be used in matrix decomposition problems [66, 57]. Lahabar *et al.* [57] compared the performance in terms of speed of SVD in MATLAB, SVD in the Intel Math Kernel Library (MKL) 10.0.4 LAPACK [65] and their implementation on a GPU using CUDA. Their test environment was an *Intel Dual Core 2.66GHz* PC and a *NVIDIA GTX 280* graphics processor. Their study focuses on evaluating the performance of parallel and serial versions of the SVD algorithms rather than some specific application of SVD. In their method, they divided the decomposition into small tasks so that each GPU thread only handled the calculations corresponding to one element of the matrix a time. We have not repeated the experiments, but the evolution of the hardware will actually reinforce the conclusion.

As the largest data set in our case is a 80×80 matrix, using a GPGPU to split the

matrix decomposition is not suitable according to the results in Table 2.4 in [57]. From this table, an SVD using a GPU will improve the performance only if the matrices are larger than $1K \times 1K$, but will impair the performance for smaller matrices. In our case, the matrices we want to decompose range from 44×44 to 80×80 which are far too small to obtain an improvement. As a result, using a GPGPU for individual matrix decompositions will not show a performance improvement in our case. Therefore, the deconvolution part is not parallelized using lower-level parallelism.

Upper-Level Parallelism

Data denoising & reorganization As for data reorganization and denoising steps, the tasks are basically matrix transformations for one single matrix — lower-level parallelism already handles that kind of task well. If we apply upper-level parallelism to these two steps, we would have to split the matrix transformation task into several sub-matrix transformation tasks. However, that would be the same as for lower-level parallelism but with an extra split and merge overhead. The larger the matrix is, the more improvement will be gained from lower-level parallelism. Therefore, the upper-level parallelism is not suitable for data reorganization and denoising steps.

Deconvolution From the medical point of view, as blood flows from one voxel to its neighbours, these voxels become related to each other. But fortunately, voxels can be considered independent in the deconvolution process (matrix decomposition), hence the perfusion-imaging analysis of each voxel is ideally parallel. This means that there is no effort required to separate the problem into a number of parallel tasks and no dependency or communication between those parallel tasks. Hence, it is possible to optimise deconvolution using upper-level parallelism.

2.2.3.3 Overall

The entire analysis contains five steps: source-image loading, data reorganization, denoising (optional), deconvolution and generation of results. The source-image loading and generation of results steps are not worth parallelising, while the remaining three steps are well suited to parallelism. Figure 2.6 illustrates how the parallelization of deconvolution step is achieved. The deconvolution step is split into sub tasks, each of which corresponds to one voxel and is assigned to one GPU thread.

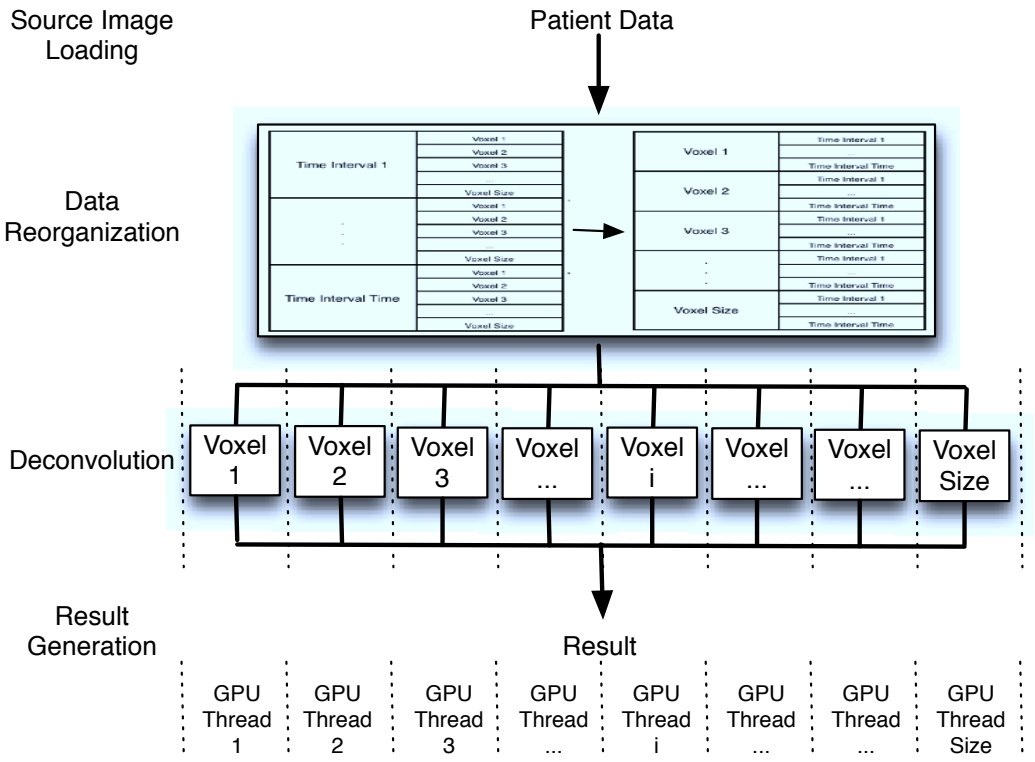


Figure 2.6: The GPGPU Parallelization Workflow

2.2.4 Parallel Perfusion Imaging Analysis

As stated in Section 2.2.3.2, the *Data reorganization* and *Denoising* steps are essentially matrix transformations and can be handled efficiently by GPGPUs. In the deconvolution step, the deconvolution of different voxels are ideal parallel tasks, so little effort is required to separate the problem into parallel tasks and there is no dependency or communication between those parallel tasks. Hence, parallel implementation can be easily achieved. The parallel algorithm for the whole workflow can then be written as Algorithm 2.

Source image loading The same as for the serial implementation, the first step in the parallel implementation is to load images into CPU memory (Line 1). The implementation of this step is exactly the same as before, so its computational complexity remains $O(Dim1 \times Dim2 \times Dim3 \times Time)$.

GPU memory copy in Line 2 is an extra step, as mentioned in section 2.1.5.1, data which will be used in the following step will be copied from CPU memory to GPU memory. The computational complexity of this step is $O(Dim1 \times Dim2 \times Dim3 \times$

ALGORITHM 2 - PARALLEL PERFUSION IMAGING ANALYSIS

```
1  CPU.A(1 : Time, 1 : Size) ← 4D MRI or CT image data
2  GPU.A(1 : Time, 1 : Size) ← CPU.A(1 : Time, 1 : Size)
3  GPU: Parallel do, shared(A, A')
4  if DoImageDenoising
5     then GPU.A'(1 : Size, 1 : Time) ← denoiseAndReorganize GPU.A(1 : Time, 1 : Size)
6     else GPU.A'(1 : Size, 1 : Time) ← reorganize GPU.A(1 : Time, 1 : Size)
7
8  GPU: Parallel do, private(i,IRF), shared(GPU.A', localAIF, CBF, CBV, MTT)
9  for n ← 1 to Dim3
10       do {
11          for i ← 1 to Dim1 × Dim2
12             do {
13                Generate localAIF(i, 1 : Time)
14                IRF ← Deconvolution result (GPU.A'(i+n × Dim1 × Dim2,1:Time) & localAIF(i, 1:Time))
15                GPU.CBF(i + n × Dim1 × Dim2) ← Max(IRF)
16                GPU.CBV(i + n × Dim1 × Dim2) ← Sum(IRF)
17                GPU.MTT(i + n × Dim1 × Dim2) ← GPU.CBV / GPU.CBF
18                }
19          }
20 CPU.CBF(1 : Size) ← GPU.CBF(1 : Size)
21 CPU.CBV(1 : Size) ← GPU.CBV(1 : Size)
22 CPU.MTT(1 : Size) ← GPU.MTT(1 : Size)
23 CBF colored map ← CPU.CBF(1 : Size)
24 CBV colored map ← CPU.CBV(1 : Size)
25 MTT colored map ← CPU.MTT(1 : Size)
```

$Time$).

Denoising and reorganization Both of the *Denoising & Reorganization* (Line 5) and *Data reorganization* (Line 6) steps can be considered as matrix transformations, which CUDA is good at handling, with a 4D input array. Each GPU thread is responsible for one element in the input array. To put it differently, each GPU thread is in charge of one and only one element which represents the intensity value of one voxel at one time interval. The thread reads the intensity value and then stores it into the right place in the target array. The task for each thread is light and the number of threads is the product of the number of voxels and the number of time points in the sampling series. The computational complexity of this step is $O(Dim1 \times Dim2 \times Dim3 \times Time)$, the same as in the serial algorithm[10].

[10]The parallel algorithm does not reduce the computational complexity compared with the serial algorithm. But hardwares for parallel implementations have a better computational power than hardwares for serial implementation (Figure 2.2).

Table 2.5: Summary of Computational Complexity

Step	Computational Complexity	Parallelism Factor
Source image loading	$O(Size \times Time)$	1
GPU memory copy in	$O(Size \times Time)$	1
Denoising and reorganization	$O(Size \times Time)$	$Size$
Deconvolution	$O(Size \times Time^3)$	$Dim1 \times Dim2$
GPU memory copy out	$O(Size)$	1
Generation of results	$O(Size \times Time)$	1

This table shows the computational complexity and the possibility of parallelism of all steps.

Deconvolution Lines 9 to 19 correspond to the most computationally expensive part of the whole workflow. The main part of each deconvolution, the decomposition of a 80×80 (or smaller) local AIF matrix, is not large enough to be parallelised using lower-level parallelism (Section 2.2.3.2). Consequently, we simply assign each decomposition to a different GPU thread. As illustrated in Figure 2.6, each GPU thread performs the deconvolution of one voxel. Therefore, hundreds of voxel deconvolutions can be performed concurrently. The overall computational complexity is $O(Dim1 \times Dim2 \times Dim3 \times Time^3)$.

In the parallel algorithm of the deconvolution step, the required input data is the output result of *Denoising & Reorganization* step which was previously computed by a set of GPU threads. So the data is already in GPU memory and there is no need to copy the result of *Denoising & Reorganization* back to CPU memory or to copy input data for *Deconvolution* step.

GPU memory copy out Lines 20 to 22 represent an extra step not present in the serial version. In this step, results of deconvolution will be copied back from GPU memory to CPU memory. The computational complexity of this step is $O(Dim1 \times Dim2 \times Dim3)$.

Generation of results The last step (lines 23 to 25), the creation of hemodynamic parametric maps, is also the same as in the serial version. The computational complexity is $O(Dim1 \times Dim2 \times Dim3 \times Time)$.

Overall The parametric maps produced by the serial and parallel implementations are identical. In other words, the quality of the results is not compromised. The computational complexity of the parallel implementation is $O(Dim1 \times Dim2 \times Dim3 \times$

$time^3$), which is the same as the computational complexity of serial implementation.

Table 2.5 illustrates the computational complexity for every step in our parallel algorithm. It also indicates the maximum possibility of parallelism[11] for each step. Section 2.2.6 will explain why *Deconvolution* step has a limited parallelism factor of $Dim1 \times Dim2$.

2.2.5 Other Parallel Implementations used for Comparison

Two other parallel approaches using OpenMP (shared-memory parallel architecture) and MPI (message-passing parallel architecture) were also implemented in the experiment. These two implementations used the same algorithmic pattern as was used for the GPGPU implementation. They both benefit from the code optimizations as the GPGPU implementation does. Furthermore, neither the OpenMP nor the MPI implementation require the memory-copy operations between CPU and GPU memory.

The same as GPU implementation, these CPU parallel implementations are optimized using data reconstruction. Unlike the GPGPU implementation, which assigns one voxel to one GPU thread, the OpenMP and MPI implementations divide all of the voxels into groups and assign one group to one CPU thread in order to reduce the CPU scheduling cost.

In order to reduce the scheduling cost, the deconvolution for different voxels are grouped together. The number of voxels is the product of the number of voxels in a group and the number of groups. Furthermore, the number of groups equals to the number of CPU threads (the number of CPU threads is determined in Section 2.3.1.1).

2.2.6 Space Complexity for Deconvolution

In principle, the amount of data transferred between the CPU memory and GPU memory should be kept as low as possible. Intermediate data structures should be created in GPU memory and freed after use without being copied to CPU memory.

Using SVD, three $time^2$ local arrays and one $time$ array are required for each voxel to store the input and output matrices. Furthermore, four $time^2$ arrays are required when calculating the inverse matrix in SVD. The memory of the input matrix can be re-used in inverse matrix calculations and the output matrices re-use the memory that

[11] The parallelism factor is defined to evaluate the maximum possibility of parallelism. A factor of 1 means this step is not parallelized in our implementation. A factor of *Size* means this step can be parallelised in *Size* threads maximum.

was allocated for calculating the inverse matrix. So the space complexity is $(time^2)$ for each voxel and $(time^2 \times$number of voxels$)$ for each scan. Due to the large number of voxels, the whole process requires a large amount of memory.

Taking a typical MRI image size (dim1 $\times$ dim2 $\times$ dim3 $\times$ number of time intervals) to be $128 \times 128 \times 22 \times 80$, with each intensity value stored in a *double* variable[12] as an example, about 200 KB[13] GPU memory is required for each voxel. Unfortunately, that exceeds CUDA's local memory limit which is 16 KB per GPU function[14]. As a result, these arrays have to be declared in global memory which leads to another problem that more than 68 GB[15] of GPU memory is required if local arrays for the whole image are to be declared simultaneously. This exceeds the overall memory (4.0 GB) available on current GPUs. Considering that local arrays are temporarily used in the deconvolution within each voxel, its space can be reused by another voxel after the first voxel's deconvolution is finished. Therefore, the solution to the memory problem is to declare a certain size of memory in global memory exclusively for local arrays to use, and assign the memory to one voxel's deconvolution and recycle it when that deconvolution is complete.

Choosing the memory size is a compromise between memory management cost and memory usage. On the one hand, declaring more memory can enable more deconvolutions to execute at the same time but it requires a large amount of memory. That reduces the effort to manage local memory but dramatically reduces the overall memory available for the rest of processes. On the other hand, if the memory size for local arrays is too small, some of the GPU cores will be idle as they are not able to allocate memory to execute.

In our experiment, the temporary memory allocations and re-allocations are performed explicitly. The size of local arrays' memory has been set to $128 \times 128 \times 4 \times 80^2 \times sizeof(double)$ which means 3GB memory is declared to cover arrays for a slice (128×128 voxels). A barrier is used after the deconvolution for each slice in order to avoid conflicting use of the temporary memory. Memory management costs can be kept at a low level as memory only needs to be re-used fewer than thirty times during the analysis. It is also large enough to cover the local memory requirement for 16,384 GPU threads, so that no GPU cores will be idle due to lack of local memory.

[12] Values in input files are stored as *short* type of variables (integer type), but we used *double* type of variables (real type) during the computation in order to reduce rounding error.

[13] More precisely for MRI $4 \times 80^2 \times 8$ bytes $= 204,800$ bytes (as four time^2 arrays are needed).

[14] Functions executed on GPU thread can only ask up to 16 KB local memory as its temporary memory

[15] $128 \times 128 \times 22 \times 200$ KB = 68.75 GB

```
        /***** Example code segment for parallel reorganization *****/
1       __global__ void reorganization( DATATYPE *d_input,
                                        DATATYPE *d_output,
                                        int time)
2       {
3               int idx = blockIdx.x*blockDim.x + threadIdx.x;
4               int i = idx % time;
5               int j = idx / time;
6               d_output[j*time+i] = d_input[i*time+j];
7       }
```

Figure 2.7: Matrix Transformation Kernel Function Fragment

Furthermore, it leaves enough memory (1GB) for the rest of the analysis.

2.2.7 Memory Bandwidth Analysis

The size of the input data is 55 MB (megabytes)[16]; for output, taking bitmap file format as an example, each voxel requires three *unsigned-char* type variables to store the RGB color information. It only costs about 1 MB[17] for each type of hemodynamic map.

As the peak memory bandwidth for GPUs exceeds 180 GB/s (since 2010), which is very fast compared to the memory bandwidth for CPUs (less than 40 GB/s), programs based on GPUs are less sensitive to data transfer rates than CPU programs. The GPU used in our experiments has a memory bandwidth of 102.4 GB/s. The cost of read/write memory can be kept as low as a few milliseconds, which only contributes a little to the overall running time.

2.2.8 Implementation Details

2.2.8.1 Matrix Transformation

Matrix transformation can apply lower-level parallelism as described in Section 2.2.3.2. Take the *Data reorganization* step as an example; each GPU thread corresponds to one matrix element in the 4D image. To put it differently, each GPU thread is in charge of one and only one element, which represents the intensity value of one voxel at one time interval. The thread reads an intensity value and then stores it to a new place in the target array. The task for each CUDA GPU thread itself (not the task it carries) is

[16] $128 \times 128 \times 22 \times 80 \times 2$ bytes (*short* data type) = 55 MB.
[17] $3 \times 128 \times 128 \times 22 \times 1$ byte (*unsigned-char* data type) = 1.03 MB.

```
        /***** Example code fragment for parallel deconvolution *****/
1       __global__ void deconvolution( INPUT *input, OUTPUT *output)
2       {
3               int idx = blockIdx.x*blockDim.x + threadIdx.x;
4               int *localmemory = find_and_pass_some_free_global_memory(idx);
5               matrix_decomposition(intput, output, idx);
6               other_matrix_operations(input, output, idx);
7       }
```

Figure 2.8: Deconvolution Kernel Program Fragment

lightweight, which means each thread has very little creation overhead. The number of GPU threads is the product of the number of voxels and the number of time points in the sampling series.

The kernel function[18] carried by a GPU thread for the transformation is simple, as shown in Figure 2.7. Line 3 gives each GPU thread a unique ID based on the CUDA indexing methodology mentioned in Section 2.1.5.2. As the input data is stored in a 1D array, threads are also identified using a 1D index. For each voxel (thread), the spatial position is calculated via Line 4 and the temporal position need to be calculated using Line 5.

In the matrix transformation, different GPU threads operate on different memory and there is no inter-thread dependencies or conflicts. There are no inter-thread communication requirement during the transformation either. Hence, the parallelization can be performed straightforward by matrix transformations in other parts of the perfusion-imaging analysis are similar to this example.

2.2.8.2 Deconvolution

The *Deconvolution* step should be parallelized using upper-level parallelism for the reasons stated in Section 2.2.3.2. Figure 2.8 shows a GPU kernel code fragment for the deconvolution step. The deconvolution task for each voxel is processed by one GPU thread. Unlike matrix transformations, each thread has a heavy computational task and the number of threads is set equal to the number of voxels. Within each deconvolution, there is a matrix decomposition and several other matrix operations[19].

[18] In CUDA programming, a CPU function is called a global function and a kernel function is the function executed in a GPU thread.

[19] It is possible to group tasks for several voxels into one single task and assign this task to one thread. However, since thread related overheads are relatively low compared with task overhead for each voxel,

The first step is also to get the identity for each thread (Line 3). For the reasons stated in Section 2.2.6, the second step in a deconvolution is to request a chunk of memory from the pre-allocated global memory and then use it as local memory, which only gets used inside the current thread (Line 4). After that, matrix decomposition and other matrix operations can be processed in the same way as they are in the regular deconvolutions (Lines 5 to 6). The local memory is released back to the global pool right after the deconvolution for the current slice is finished.

During the deconvolution, different threads operate on different voxels and there is no data dependency between different threads. As a result, no synchronization is required in the body of the deconvolution step[20]. However, different GPU threads share the same memory pool, which results in a small extra data dependency between different threads. Fortunately, since the dependency only happens at the start of each deconvolution, our local memory management (stated in Section 2.2.6) ensures that all the threads can execute without waiting for local memory. Another reason that keeps the overhead low is that the memory of all local arrays we assigned is relatively large compared to the memory of the local array needed by one voxel.

2.2.9 Results Check

Since the serial and parallel implementations perform the same task, results (hemodynamic maps) derived from them should be identical. In order to double check the correctness of results, a test is performed. It compares the bitmaps delivered by different implementations and its results confirm that the hemodynamic maps are almost identical with minor differences.

The differences are caused by rounding error. In CUDA (parallel implementation), the rounding mode used is *round to nearest, ties to even*, while GCC (serial implementation) uses a rounding mode of *round towards zero*. The different rounding modes may cause slight different values[21] in the hemodynamic maps derived from serial and parallel implementations. However, these differences are not visually apparent in the hemodynamic maps, so we still consider the results to be identical.

---

it is not necessary to group them.

[20] Except for that performed implicitly behind the scenes for the memory allocation and de-allocation calls.

[21] The difference rarely happens. The odds of it occurring is less than 0.01% during your experiments.

Table 2.6: Devices used in the Experiments

CPU	GPU
Two Intel®Xeon®E5620 CPUs	One Tesla C1060 GPU
Dual cores each (4 cores in total)	240 cores
3 GHz each core	1.44 GHz each core
8 GB global memory each processor	4 GB global memory each processor
4 MB cache each processor	16 KB shared memory each processor
2.4 GFLOPS each processor	933.12 GFLOPS each processor

This table shows the CPUs and GPU used in our experiments. The multi-core CPU program runs on a four-core node, which consists of two dual core CPUs, while the GPGPU program runs on a single GPU.

2.3 Experimental Results

In this section, the performance of serial, GPGPU parallel and other CPU parallel implementations are presented.

2.3.1 Experimental Environment

2.3.1.1 CPU Environment

The CPUs used in our experiments are two $Intel(R)Xeon(R)$ CPUs, each of which contain two cores. The frequency of each CPU core is 3 GHz. The overall CPUs memory is 8 GB and their cache size is 4 MB each. The serial code is compiled using GCC 4.1.2, while OpenMPI 1.4.2 has been used for MPI programming [48] and Intel compiler (version 11.0) has been used for OpenMP programming [46]. The serial, OpenMP and MPI implementations were all executed on the CPUs. The serial version of the perfusion-imaging analysis only exploits one core in the CPUs. Both the OpenMP and MPI implementations use all of the four cores in the two dual-core CPUs. Furthermore, since there are only four cores in total, the number of threads are set to four in both of the OpenMP and MPI implementations.

2.3.1.2 GPU Environment

The GPU used in the experiments is a Tesla C1060 GPU which provides 240 GPU cores in total. The frequency of each GPU core is 1.44 GHz. The GPU's single precision floating point performance (peak) is 933.12 GFLOPS and it has 4 GB of global memory and 16 KB of shared memory. CUDA 4.0.2.1221 by NVIDIA, released in May 2011, has been used as the programming platform.

Since a GPGPU program requires a CPU thread to act as the 'master' thread, the GPGPU implementation runs on the Tesla GPU that cooperates with the same CPU as the serial implementation. The master CPU thread only deals with lightweight tasks and contributes a tiny fraction to the overall computational task (*Deconvolution* Step).

2.3.1.3 CPU and GPU Environments Comparison

As mentioned in section 2.1.4, it is not sensible to evaluate the performance using CPUs and GPUs with the same single-precision, floating-point performance. Based on our research in June 20^{th}, 2012, the Intel®Xeon®E5620 CPU cost approximately £500 each [67], which is £1,000 for the two CPUs; the Tesla C1060 GPU has a cost of £950. Both these CPU and GPU are commodity hardware, which are widely used. The two hardware devices can be considered as almost equivalent in price and the performance improvement using multi-core CPU and GPU based on these hardware devices are compared directly.

The features of the hardware equipment used for the experiments are displayed in Table 2.6.

2.3.1.4 Test Datasets

One of the test datasets we used is composed of simulated images, each containing $128 \times 128 \times 22$ voxels, and the number of time intervals is 80, which is one of the sizes of MRI images. Another test data set in the experiment consisted of $128 \times 128 \times 11$ voxels with 44 time intervals, which is one of the sizes of CT images. Input data is stored using the *short* data type, which requires 2 bytes for each element. The results shown below are the arithmetic mean of ten repeated tests. The bottleneck of the whole process is the matrix decomposition, whose computational complexity is only affected by the size of the matrix. Based on our experiments, the running times for our simulated data and patient data are the same. The only factor that matters is the size of the images. The use of simulated data does not affect the performance.

2.3.2 Performance for Each Step

Table 2.7 shows our measurements of the performance for each step in the whole workflow for the size of an MRI dataset ($128 \times 128 \times 22 \times 80$).

The steps *Source image loading* and *Draw parametric maps* are not suitable for parallelization and only have serial implementations. These two steps are not parallelized

Table 2.7: Performance for Each Step

Step	Serial Running Time (s)	GPGPU Running Time (s)	Speedup Factor
Source image loading	0.10	0.10	–
Data copying (CPU to GPU)	Not Applicable	0.17	–
Data reorganization	1.1	0.01	110
Denoising & reorganization	4.3	0.01	430
Deconvolution	2.11×10^3	5.64×10^2	3.74
Data copying (GPU to CPU)	Not Applicable	0.01	–
Draw parametric maps	0.20	0.20e	–
Total	2.11×10^3	5.64×10^2	3.75

This table indicates the processing time for both the serial and parallel algorithms for each individual step. The running time displayed in this table is the average running time of ten repeated tests, where each test deliveries almost the same results with a standard deviation less than 1%. Because of *Source image loading* and *Draw parametric maps* steps are not suitable to be parallelized and *Data copying* steps only happen in the parallel algorithm, speedup factors are not calculated for these steps.

in any of our parallel implementations and are processed serially in the 'master' thread. Their running time is the same as the running time in the serial implementation.

In parallel deconvolution, the first step of the parallel workflow is to copy data from CPU memory to GPU memory. The input data is about 220 MB, which is mainly an array with $128 \times 128 \times 22 \times 80$ *short* elements. The copying takes 0.17 seconds. The size of the results to be moved back from GPU memory to CPU memory is much smaller and only takes 0.01 seconds to perform the copy back operation.

In serial deconvolution, the *Denoising & Reorganization* step, prior to deconvolution, takes 4.3 seconds compared to the 1.1 seconds for reorganization only. After applying parallelization to these steps, the running time dramatically decreased to 0.01 seconds. The speedup factors are 430 and 110, respectively.

The running time of the *Deconvolution* step, the most expensive step of perfusion-imaging analysis, was reduced from 2108 seconds to 564 seconds after applying parallelization. The speedup factor is 3.74. However, it still dominates the running time.

This result is consistent of the computational complexity analysis mentioned in sections 2.2.2 and 2.2.4.

Since the OpenMP and MPI implementations share the same *Source image loading*, *Data reorganization*, *Denoising & reorganization* and *Draw parametric maps* steps with the serial implementation and do not have the memory copy steps, their performance is only evaluated in the overall performance section (Section 2.3.3).

Table 2.8: Overall Performance

Data Size ($Dim1 \times Dim2 \times Dim3 \times time$)	Serial Running Time (s)	GPGPU Running Time (s)	OpenMP Running Time (s)	MPI Running Time (s)
$128 \times 128 \times 22 \times 80$ (MRI Image Size)	2114	564 Speedup Factor = 3.75	956 Speedup Factor = 2.21	619 Speedup Factor = 3.42
$128 \times 128 \times 11 \times 44$ (CT Image Size)	360	65 Speedup Factor = 5.56	159 Speedup Factor = 2.26	94 Speedup Factor = 3.84

This table indicates the overall running time and speedup factor for all of the serial and parallel implementations.

2.3.3 Overall Performance

Table 2.8 shows the overall speedup improvement gained from GPGPU. As shown in Table 2.7, the *Deconvolution* step is the slowest step, which takes hundreds of times longer than the other steps. Hence, the overall performance should be closely related to the performance of *Deconvolution*. As a result, although other steps can be improved by large factors, the overall running time can be roughly considered as the same as the running time for the *Deconvolution* steps which can also be found in Table 2.7. In other words, the final performance depends on the *Deconvolution* step and the overall speedup factor is 3.74, which is very close to 3.75 from *Deconvolution* given in Table 2.7.

Lorenz *et al.*[35] did experiments on deconvolution using local AIFs. They did performance experiments on a small data set size, which was 128×128 voxels per slice, 11 slices and the number of time intervals was 44, one of the typical CT image sizes. The overall running time to finish their deconvolution is still six minutes (the same as in our experiments) with a speedup factor of 5.56. This is reduced to one minute and 5 seconds after applying parallelism. However, in MRI images, the data size has increased to 128×128 voxels per slice, 22 slices and the number of time intervals is now 80, approximately four times as much data. It takes about 35 minutes in our serial implementation.[22]

The OpenMP CPU parallelization provides speedup factors of 2.21 and 2.26 for the the MRI image size data and CT image size data, respectively. CPU parallelization using MPI leads to a better performance compared with OpenMP, which results in speedup factors of 3.42 and 3.84, respectively.[23] For MRI image size data, the GPGPU

[22] It would take around 40 minutes using Lorenz's methods by computational complexity estimation.

[23] Considering that the number of CPU cores in the experiments is four, both of the two CPU parallel methods have a theoretical upper boundary of performance improvement factor of 4.

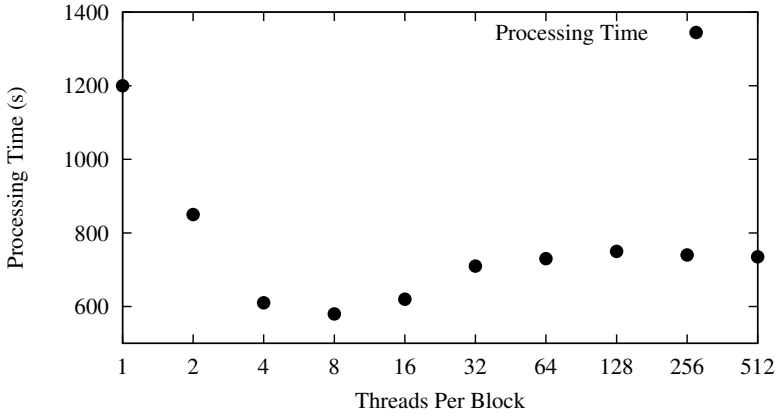


Figure 2.9: Threads Per Block

This figure shows the relationship between the parameter *Threads Per Block* and processing time. Note that the X-axis is in logarithmic (base 2) scale.

approach takes 59% of the time of the OpenMP approach and 91% of the time of MPI approach. For CT image size data, our GPGPU approach has more than double the performance of the OpenMP method and has 1.45 times performance of the MPI method. Thus, for both of the MRI image size data and CT image size data, our GPGPU parallel implementation shows a better performance than CPU parallelization.

2.3.4 GPGPU Parameters

Figure 2.9 shows that performance changes with the number of threads per block, as discussed in Section 2.1.5.2. In our experiments, eight threads per block provide the best performance. According to the design of CUDA, all of the threads of a block should be assigned to the same processor core. As the total number of threads is stationary, if the number of threads per block is too small, it will lead to a large number of blocks and therefore lead to extra scheduling overheads. On the other hand, the tasks for each thread are very heavy. Hence, the best performance is not achieved at 128 or 256 threads per block but with a smaller number. We surmise that the fall in performance with increase of threads per block is due to memory resource contention or scheduling overheads.

Furthermore, since each GPU thread has a similar workload and the total number (*Size*) of GPU threads is relatively large compared with the number of threads per block, load balancing is not an important performance factor when changing the number of threads per block.

2.4 Conclusion

In this chapter we introduced a parallel implementation of perfusion-imaging analysis which provides considerable speed improvement and the same quality of results compared with current serial implementation.

We have analyzed computational complexity and feasibility for every individual step in the perfusion imaging processing and applied different parallelism methods based on the analysis. Different parallelism is applied for different steps based on their features. *Data reorganization* and *Denoising & reorganization* steps use lower-level parallelism in which each GPU thread is in charge of the task for one voxel and one time point. In the *Deconvolution* step, upper-level parallelism is used, by distributing deconvolution (contains all of the time points) for different voxels to different GPU threads. Theoretically, for both of these steps, parallelization enables all of the voxels in a scan to execute concurrently. However, due to the limitation of local memory, parallelization for the *Deconvolution* step can only be performed one slice at a time.

The *Deconvolution* step is the bottleneck for perfusion-imaging analysis, although the speedup factor is more than one hundred for both the *Data reorganization* and *Denoising & reorganization* steps, the overall performance speedup factor is limited by this bottleneck. The overall processing time is reduced from six minutes to 65 seconds for our CT test dataset and from 35 minutes to less than ten minutes for our MRI test dataset. The performance speedup factors are 5.56 and 3.75, for CT and MRI images respectively. Meanwhile, the quality of serial and parallel output images remains unchanged. The speedup also depends on the CUDA configuration parameters which determine how tasks are assigned to GPU cores. Our experiments also show that the four core CPU parallel implementation using OpenMP and MPI gains a speedup of 2.21 to 3.84 depending on the data size, which is smaller than the improvement brought about by GPGPU implementations. GPGPU implementation is superior to serial implementation and to both CPU parallel implementations.

As time is vitally important in clinical diagnosis, especially for acute stroke cases, the earlier we deliver the result for diagnosis, the less damage will be caused by strokes and the higher the possibility that treatment will be effective. Therefore, performance is as important as accuracy in perfusion imaging, and our implementation shows the potential of GPUs for speeding up clinical diagnosis. Our implementation using GPGPU can significantly reduce analysis processing time based on local AIFs, which increases the possibility of local AIFs being used in clinical diagnosis.

Furthermore, with the improvement of CT and MRI imaging, the size of input images is likely to increase, which will definitely increase the demand for speedup by exploiting parallel hardware. The required speed up will not come from individual CPUs getting faster as their speed is reaching limits of current technology. However, based on the current trend in performance improvement for GPUs, it is highly possible that GPUs will provide the computational power we will need.

Chapter 3

Noise Reduction Using Gaussian Process Regression

[Parts of this chapter have been published as 'Computer tomography perfusion imaging denoising using Gaussian process regression' in Physics in Medicine and Biology, 2012 [40].]

3.1 Introduction

In this chapter, we propose Gaussian process regression [68, 69] based approaches which make use of temporal information in perfusion source images to reduce noise. We compare the effectiveness of other perfusion imaging noise reduction methods with the effectiveness of our methods in terms of the quality of tissue time-concentration curves, contrast-to-noise ratio and the quality of the resulting hemodynamic quantity maps. Our Gaussian process regression based methods are designed to handle the high level of noise present in CT perfusion images, but they also show noticeable effects in MRI images which have a lower level of noise.

3.1.1 Motivation

Since X-ray radiation increases the risk of inducing cancer [20, 21], CT scanning is constrained by a tradeoff between image quality and the amount of radiation exposure to the patient. In order to reduce the risk, CT scans are relatively short and the radiation dose level is low. Therefore CT perfusion imaging suffers from low contrast to noise ratio (CNR) making post-processing of the acquired images (to produce perfu-

sion parametric maps) problematic. The noise leads to difficulty when attempting to estimate parameters, such as cerebral blood flow (CBF), cerebral blood volume (CBV) and Tmax [70]. The quality of these hemodynamic maps decreases dramatically with the increase of noise level [71]. The majority of current noise reduction approaches for CT images are designed for 3D (spatial only) information, including spatial decimation (weighted mean filters, Gaussian filters [72, 73, 17]), techniques based on wavelet transforms [74, 75, 76] and curvelet transforms [77, 78, 79]. Another way to suppress the noise is to use regularization techniques, for example, truncated SVD [26, 27], during deconvolution. However, all of these methods were designed for CT and MRI imaging. The sequence of scans needed for perfusion imaging opens up the possibility of using coherence in the time domain to enhance the signal-to-noise ratio.

Perfusion imaging contains temporal information so that the tissue time-concentration curves are expected to be continuous (in time) and follow a specific pattern. Hence, there is potential to reduce the noise level based on the temporal information. On the one hand, as mentioned in Section 1.8, there is a truncation step which reduces the impact of noise. The truncation step can be considered as noise reduction on the AIF matrices[1]. It improves the quality of hemodynamic quantification results by operating on AIF matrices, that are one of the two inputs to deconvolution. On the other hand, the tissue time-concentration curve, the other input of a deconvolution, also has potential for denoising. Furthermore, since the AIF matrices are derived from an artery but tissue time-concentration curves can be from any part of the brain, the signal to noise ratio in an AIF is expected to be greater than it is in other tissues' time concentration curves[2]. Noise reduction in the tissue time-concentration curves is more promising than in AIF.

Since CT images have a high spatial resolution, which is usually 512×512, if the resolution is reduced to a lower level such as 128×128, they still have a reasonably sharp resolution. As a result, spatial decimating (3D image) based filters, which reduce noise as well as resolution, are also acceptable noise reduction methods. Therefore, we have also developed a method using both temporal and spatial information, which works as a combination of temporal noise reduction filter and spatial mean filter.

[1] It can also be considered as noise reduction based on temporal information since the AIF matrices are generated using an AIF that is based on temporal information.

[2] The AIF curves are chosen from voxels with a large signal (as stated in Section 1.9), and the noise is approximately uniform. So the signal-to-noise ratio is large.

3.1.2 Imaging Denoising Filters

In imaging processing, filters are used either to smooth the image or to enhance and detect edges in the image. Since noise is a notable problem in all kinds of images (not just medical images), along the years numerous noise reduction filters have been introduced. Image denoising filters can be catalogued by their target applications. Some of the methods are designed for all kinds of images, such as weighted mean filters, median filters and bilateral filters, while others are already adjusted specifically for medical images by previous research, such as wavelet filters and curvelet filters[3]. Denoising filters can be also catalogued by the range of their denoising kernels. Most of the noise-reduction methods are based on one image volume[4] from a single sampling time point[5], called spatial (decimating) based filters, including weighted mean filters, bilateral filters, wavelet filters and curvelet filters; while a minority of them are based on all the volumes from the whole sampling series, such as TIPS filters.

In this section, denoising filters were described using a 2D example, where 'pixel' replaces 'voxel' for simplicity[6]. These methods can also be applied in a 3D domain[7].

3.1.2.1 Weighted Mean Filter

The idea of weighted mean filtering is to replace each pixel value with a weighted mean value of its neighbours, including itself, which is similar with mean filter. Instead of using the same weight for all of the pixels in each kernel (the target pixel plus its neighbours), which leads to all of the neighbours contributing equally to the final value, the weighting function is determined by an approximation of the Euclidean distance between each neighbouring pixel and the target pixel, as a result, the closer a pixel is to the target pixel, the more it will contribute to the final values.

A weighted mean filter can also be considered as a convolution filter. It is based around a kernel near the target pixel, so the neighbours can be sampled when calculating the mean. This convolution is repeated until it reaches all of the pixels in the

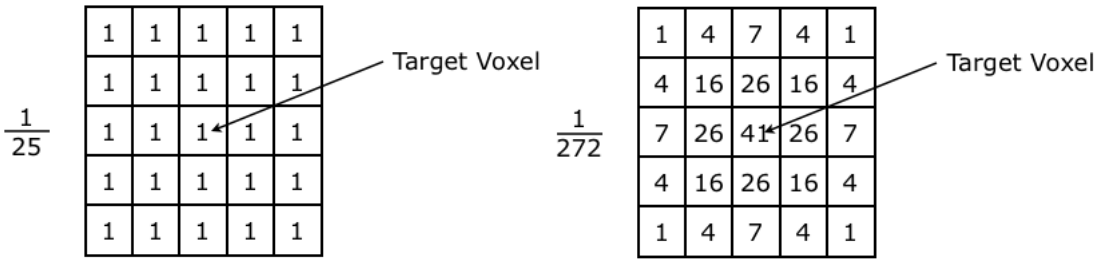
[3] Wavelet and curvelet filters can be applied to all kinds of images.

[4] To make the explanation easier to understand, in this chapter, the term, 'volume', refers to a 3D image (consisting of all of the 2D slices acquired in one single sampling time point) but not one 2D slice, which is different from the convention. In other words, data acquired in each PCT scan is in the form of a series of volumes.

[5] Unsurprisingly, since in many medical applications there is only a single image (volume) not a time series.

[6] Since the filters are designed for regular images.

[7] Though only 2D denoising filters are used in this chapter. The reason for that is stated in Section 3.1.4.



(a) Mean filter (b) Gaussian Filter

Figure 3.1: Weight Functions in Weighted Mean Filters

This figure shows two example of the weight functions of weighted mean filters. $\frac{1}{25}$ and $\frac{1}{272}$ are the normalization factors which are the reciprocal of the sum of all the weighting values.

image[8]. Figure 3.1a is an example of mean filter with a 5×5 kernel.

Spatial weighted mean filtering is simple to understand and easy to implement, its execution overhead is also low due to its simplicity. However, the main problem with spatial weighted mean filtering is that if the target pixel is in the boundary (edge) area between two signal levels, the filtering will interpolate new values for pixels on the edge and thus blur the edge.

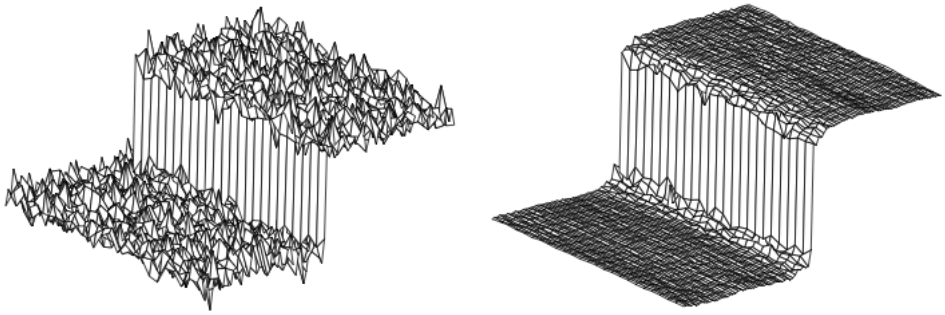
Gaussian filter, also known as Gaussian smoothing, is a type of weighted mean filter. It uses Gaussian distribution to determine its weight function (Figure 3.1b). The figure indicates that the target pixel has the highest weight of all of the pixels.

3.1.2.2 Bilateral Filter

Bilateral filter is a smoothing and edge preserving filter which was introduced by Tomasi *et al.* [80]. It can be considered as an extension of weighted mean filter with two factors contributing to the weighting function between two pixels: space weight and similarity weight. The space weight represents the spatial closeness of the pixels and the similarity weight indicates how the intensity values of these two pixels are similar to one another.

Figure 3.2 shows an example of how bilateral filters preserve the edge while they reduce the noise. The bilateral filtering processed results (Figure 3.2b) illustrate two of their features: (1) in a smooth region, bilateral filtering is similar to standard domain filtering, such as weighed mean filtering, which uses averaging to remove the

[8]The situation at the boundary (the most outer) pixels are slightly different since they have less neighbour pixels. Usually, a boundary pixel has a shrinking kernel to keep every kernel full, or is simply skipped.



(a) Raw (b) Denoised

Figure 3.2: Bilateral Filtering Example

These two figures (from [80]) are an example of how bilateral filtering works. The one on the right shows how bilateral filtering recreates the sharp boundary between a dark and a bright regions.

small differences between pixels caused by noise; (2) at a boundary (edge) region, bilateral filtering is still a standard domain filtering but some pixels in the domain will be ignored as determined by the similarity weight. The bilateral filter used in the experiments will be illustrated in Sections 3.1.2.4 and 3.2.4.1.

3.1.2.3 Wavelet Filter

Wavelets were first developed by Grossmann *et al.* in 1984 [81]. A wavelet transform is similar to a Fourier transform in which signals are represented as a sum of subsignals with given features. The main difference between a wavelet transform and a Fourier transform is that a wavelet transform is localized in both time and frequency domains while a Fourier transform is only localized in frequency. Wavelet filters have been used previously for medical-image noise reduction [74, 82].

The basic idea of wavelet denoising filtering is a transformation that splits the signal into different scale components, where a frequency range can be assigned to each scale component. After that, different filters, such as high-pass filters and low-pass filters, will be applied to different components in order to remove different types of noise. For example, to remove white noise, the focus can be put on components with high frequency. After reducing the noise in each component, a wavelet denoising filter combines all of the components together to generate the denoised image.

3.1.2.4 TIPS Filter

Mendrik *et al.* [83] introduced a bilateral filtering based method called time-intensity profile similarity (TIPS). The TIPS filter is essentially still a bilateral filter. Compared with the regular bilateral filter, the TIPS filter has a similar space weight function but a much different similarity weight function. The weight function in regular bilateral filter is only calculated by comparing two single values, while in TIPS filter, it is evaluated by comparing two curves. More specifically, the space weight function is defined as:

$$c(t,x) = \exp\left(-\frac{1}{2}\left(\frac{d(t,x)}{\sigma_d}\right)^2\right) \tag{3.1}$$

where x is the target voxel[9] and t is its neighbour, $d(t,x)$ is the Euclidean distance and σ_d determines which distance is considered to be close.

The way TIPS filter determines the similarity weight function is specific to medical perfusion imaging, in order to exploit the features of perfusion imaging and to make use of previous experience. The similarity weight function is defined as:

$$s(t,x) = \exp\left(-\frac{1}{2}\left(\frac{\zeta(t,x)}{\sigma_r}\right)^2\right) \tag{3.2}$$

where $\zeta(t,x)$ measures the intensity difference between *voxelx* and *voxelt*, σ_r also determines which difference in intensity value is considered to be similar. Since for each voxel in the perfusion images, there is a corresponding tissue time-concentration curve, $\zeta(t,x)$ measures the intensity difference via the sum of squared differences value between *voxelx* and *voxelt*.

The σ_d and σ_r are important as they determine the size of the denoising kernel and the level of smoothness. They are determined based on the features of PCT images and fixed in different scans. More details of the setting of these two parameters will be illustrated in Section 3.2.4.1. For each pair of spatial (3D) positions, the weight function only has to be determined once and is then applied to volumes at all the sampling time points.

If we consider TIPS filter as a variant of weighted mean filter, TIPS filter uses the temporal information to determine its weight function and has an spatial-based convolution kernel. As a result, it is still a spatial filter rather than a temporal filter.

[9] Since TIPS filter is designed for perfusion imaging, voxel is used instead of pixel.

3.1.3 Characterization of Noise in CT Imaging

Two important characteristics of CT imaging that affect image quality are blur and noise [8]. The blur effect limits the ability to see small objects, boundaries of different tissue types and other image details. The blur effect may be diminished by reducing the size of voxels, by using reconstruction filters, by modifying the focal spot size and detector size or by applying edge enhancing filters. The noise reduces the visibility of low contrast objects. However, reducing the size of voxels or absorbing fewer photons decreases the blur effect but increases the noise effect; increasing the size of voxels or applying a smoothing filter can reduce the noise effect but it leads to more blurred results; increasing the radiation dose can reduce the noise but increase the risk of inducing cancer. It is a trade-off to find the balance between reducing blur effect, reducing noise effect and avoiding harm to patients.

As each voxel in a CT image is acquired by adding values from a number of different projectors together and there is an individual noise in the data acquired from each of these projectors, the final noise in each voxel is the combination of individual noises so it follows a normal distribution [84].

3.1.4 Spatial resolution of CT imaging

CT images acquired usually have 16 slices (the third dimension). However, due to the low signal-to-noise ratio in CT images, these slices are too noisy. In order to improve the signal-to-noise ratio, 8 slices of CT images are combined into 1 slice during preprocessing by taking the average intensity value of voxels with same 2D positions. This operation reduces the number of slices to 2.

Due to this operation, the distance between adjacent voxels within the same slice (1mm - 2mm) is much smaller than the distance between adjacent voxels from different slices (10mm or more). This is the reason why 2D denoising filters, instead of 3D denoising filters, are used in this chapter.

3.2 Methods

3.2.1 Gaussian Process Regression

A Gaussian process is a generalization of the Gaussian probability distribution. Whereas a probability distribution describes random variables which are scalars or vectors, a

stochastic process governs the properties of functions. A Gaussian process is a collection of random variables, any finite number of values that have a joint Gaussian distribution, and is widely used to solve regression problems [69]. As stated in Section 3.1.3, the noise in CT perfusion imaging can be considered Gaussian, which fits the assumption of Gaussian process regression, thus the CT perfusion images can be denoised properly using Gaussian process regression.

Gaussian process regression was designed to be a prediction tool. It is usually used to predict the unknown dependent variable for any given independent variables based on known but noisy observations of the dependent and independent variables. Compared to the least squares method which fits linear regressions, Gaussian process regression can be used to fit polynomial, non-polynomial or even arbitrary regression problems by adjusting its model parameters. Gaussian process regression is a less parametric method, as it does not match its target function to some specific models (e.g. linear, quadratic or cubic models). "A Gaussian process can represent the target function obliquely, but rigorously, by letting the input data 'speak' more clearly for themselves", said by Ebden [85]. However, it still needs a few parameters to fit the given assumptions about the predicated function.

The key idea of our method is to consider the input intensity values as noisy observations (noisy dependent variables) of a function and predict all of the given noise free dependent variables on the same independent variables in order to reduce the noise, the details of which will be stated below. A Gaussian process regression is specified by its mean and variance. In our algorithm, the denoised value is considered as the expected mean value and its variance is used to calculate the confidence interval.

In the Gaussian process regression for noisy observations, the function values themselves are not accessible and only the noisy observations are available:

$$Y = f(T) + \varepsilon \tag{3.3}$$

where $T = \{t_1, t_2, \cdots, t_n\}$ is an input vector which is the time series in this case, f is the function, $Y = \{y_1, y_2, \cdots, y_n\}$ are noisy observation values and ε represents Gaussian noise with variance μ_n^2.

In the medical perfusion images, only the noisy intensity values, not the actual intensity values, are accessible. This corresponds to the same situation with Gaussian process regression for noisy observations.

The function that measures how one observation relates to another in a Gaussian process is a covariance function $k(t_p, t_q)$, where t_p and t_q are time series. Generally,

if $t_p \approx t_q$, $k(t_p,t_q)$ will approach its maximum, indicating $f(t_p)$ is almost perfectly correlated with $f(t_q)$, as we expect when the function is continuous and smooth. On the other hand, if t_p is distant from t_q, $k(t_p,t_q)$ will be very small, indicating no linear relationship between $f(t_p)$ and $f(t_q)$.

To estimate f_*, the expected mean value of $f(t_*)$, for an arbitrary t_* in equation 3.3, three matrices are required: $K(T,T)$ in Equation 3.4 is used to define the correlation between independent variables; $K(T_*,T)$ in Equation 3.5 is used to measure the covariance between observation and unknown points; $K(T_*,T_*)$ in Equation 3.6 is the covariance between unknown points:

$$K(T,T) = \begin{bmatrix} k(t_1,t_1) & k(t_1,t_2) & \cdots & k(t_1,t_n) \\ k(t_2,t_1) & k(t_2,t_2) & \cdots & k(t_2,t_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(t_n,t_1) & k(t_n,t_2) & \cdots & k(t_n,t_n) \end{bmatrix} \tag{3.4}$$

$$K(T_*,T) = \begin{bmatrix} k(t_*,t_1) & k(t_*,t_2) & \cdots & k(t_*,t_n) \end{bmatrix} \tag{3.5}$$

$$K(T_*,T_*) = [k(t_*,t_*)] \tag{3.6}$$

The two key terms for Gaussian process regression can be calculated as:

$$f_* = K(T_*,T)K(T,T)^{-1}[y_1,y_2,\cdots,y_n]^{transpose} \tag{3.7}$$

$$var(f_*) = K(T_*,T_*) - K(T_*,T)K(T,T)^{-1}K(T_*,T)^{transpose} \tag{3.8}$$

where $var(f_*)$ is its variance. Using this formula to calculate f_* for whole time series, we can reconstruct $f(x)$ in Equation 3.3 with its variance function $var(f)$.

3.2.2 Denoising Using Gaussian Process Regression (GPR)

In our noise reduction method, instead of considering the data as a sequences of images, the input data is treated as a group of signals. Each signal is a tissue time-concentration curve whose data is obtained across the whole time series and the number of signals equals the number of voxels in images on one sampling time point.

The key rationale for using Gaussian process regression (*GPR*) to denoise CT images relies on the continuity of tissue time-concentration curves (temporal information). Our GPR works as a signal-based noise-reduction method much more than a image noise reduction method. The sampling time is the independent variable and the

intensity value is the dependent variable. So the first step is to obtain the 'signal', the tissue time-concentration curve from a fixed voxel, consider it as a noisy observation, and then use the observed y_i to build vector Y in equation 3.3, where t_i is the i_{th} sampling time in the time sequence and y_i is the noisy observation for t_i.

Typically the covariance functions which are used in the denoising algorithm have parameters with the assumed form [69]:

$$k(t_p,t_q) = \mu_f^2 exp(-\frac{1}{2l^2}(t_p - t_q)^2) + \mu_n^2 \delta_{pq} \tag{3.9}$$

where l is the length-scale, μ_f^2 is termed the signal variance and μ_n^2 is the noise variance (ε in Equation 3.3). δ_{pq} is a Kronecker delta which equals to one if $p = q$ and equals to zero otherwise. Generally, the closeness between the time point of the target intensity value and the time point of a reference intensity value decrease with the increase of the distance of time points between the target and the reference. The covariance function has a maximal value of $\approx(\mu_f^2 + \mu_n^2)$ when the target and the reference are for the same time point; while it also has a minimal value of ≈ 0 when the target and the reference's sampling time points are distant from each other. More details about parameters and their selection will be given in section 3.3.2.

According to Equations 3.4, 3.5 and 3.7, for each voxel in the tissue time-concentration curve, the Gaussian process regression filtering equation is defined as follows:

$$f'(t_x) = \begin{bmatrix} k(t_x,t_1) & k(t_x,t_2) & \cdots & k(t_x,t_n) \end{bmatrix} \begin{bmatrix} k(t_1,t_1) & k(t_1,t_2) & \cdots & k(t_1,t_n) \\ k(t_2,t_1) & k(t_2,t_2) & \cdots & k(t_2,t_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(t_n,t_1) & k(t_n,t_2) & \cdots & k(t_n,t_n) \end{bmatrix}^{-1} \begin{bmatrix} f(t_1) \\ f(t_2) \\ \vdots \\ f(t_n) \end{bmatrix} \tag{3.10}$$

where t_i is the sampling time of the i_{th} time point, $f(t_i)$ is the observed noisy (raw) intensity value and $f'(t_i)$ is the denoised intensity value for the t_i time point. The confidence interval, which indicates the certainty of the estimated value, is not used in our method although it can be determined at the same time.

For each voxel, Equation 3.10 needs to be applied for each time point, then t_x needs to iterate through t_1 to t_n, to reconstruct the tissue time-concentration curve point by point. Furthermore, in order to reach all the noisy values in the images, the denoising filter should go though all of the voxels in the brain. So the Equation 3.10 should be repeated $(number\ of\ voxels \times number\ of\ time\ points)$ times. Since the k_{th} sampling-time value in different tissue time-concentration curves are the same, because they are from the sample volume (in terms of the temporal axis not the 3D axis), t_k is the same for different voxels. As a result, all of the voxels have the same covariance matrix

(Equation 3.4) and also the same inverse matrix of the covariance matrix in Equation 3.10. So the inverse matrix only needs to be calculated once and can then be reused for each scan in order to reduce the computational complexity.

3.2.3 Denoising Using Multiple Observations Gaussian Process Regression (MGPR)

Since blood always flows from one voxel to the neighbouring voxels, the intensity values in the adjacent voxels are expected to be similar at any time point. In other words, the intensity value should also be continuous in the 3D source images. An exception happens at the boundaries of different tissue types, such as the boundaries of blood vessels. Typically, a CT brain image has a low contrast-to-noise ratio but a high spatial resolution (usually 512×512 per slice). Hence, a both spatial and temporal based noise reduction method called multiple observation Gaussian process regression (MGPR) is also investigated. First of all, images are divided into many small blocks of voxels and each block is called a kernel. After that, in a manner similar to other spatial based methods, in MGPR, a new tissue time-concentration curve of the target voxel is calculated from all of the voxels in the same kernel. These adjusted voxels within the same kernel will be treated as multiple noisy observations in a Gaussian process regression. Thus the key idea of MGPR is that each sampling time point of any voxel has been observed multiple times instead of once.

For example, if the decimation kernel size is 2×2, the input vector will be built as follows:

$$T = \{t_1, t_1, t_1, t_1, t_2, t_2, t_2, t_2, \cdots, t_n, t_n, t_n, t_n\} \tag{3.11}$$

$$Y = \{y'_1, y''_1, y'''_1, y''''_1, y'_2, y''_2, y'''_2, y''''_2, \cdots, y'_n, y''_n, y'''_n, y''''_n\} \tag{3.12}$$

where t_i is still the sampling time and $y'_i, y''_i, y'''_i, y''''_i$ denote four voxels in the same 2×2 kernel at time point i. The multiple observation Gaussian process regression filtering equation is the same as Equation 3.10 but with input vectors four times larger than GPR.

The decimation factor can be 1×1, 2×2, 3×3 or even larger. The GPR method mentioned in section 3.2.2 corresponds to the MGPR with a kernel size of 1×1. The MGPR used in our experiments has a decimation factor of 3×3. Furthermore, MGPR uses the same covariance functions and parameters as our GPR (described in Section 3.3.2). The main advantage of MGPR is that it makes use of the continuity in both of the spatial and temporal information. However, it can not handle the boundaries of

different tissue types well as it blurs their boundaries when they lie within a block of voxels, contributing to one signal.

As described in Section 3.1.4, the continuity in the adjacent voxels from different slices is much weaker than it is in the adjacent voxels from the same slice. Thus, a 2D kernel, instead of a 3D kernel, is used in the experiments.

A disadvantage of using MGPR is that all the voxels within a kernel have to be treated in the same way, so if the kernel is large, distanced voxels (to the target voxel) contribute the same weight as close voxels (to the target voxel). This also blurs hemodynamic maps.

3.2.4 Other Methods for Comparison

In this sub section, we will introduce three noise reduction methods as comparison methods. Since results from spatial decimation based methods have a different resolution with results from raw images and GPR methods, the criteria contains two factors: quality and resolution. Results from all these methods are compared using both statistical methods and user trials. For statistical methods, examination is hard as there is a trade-off between statistic parameters (a parameter quality) and resolution. For user trials it is easier, since both quality and resolution information is contained by the hemodynamic maps. More details can be found in Section 3.3.

3.2.4.1 TIPS Bilateral Filter

As mentioned in Section 3.1.2.4, Mendrik *et al.* [83] have recently developed a bilateral filter based method called Time-Intensity Profile Similarity (TIPS) that produces higher quality CBF maps than Gaussian filters (Section 3.1.2.1), 3D bilateral or 4D bilateral filters[10]. In their method, they also make use of temporal information. However, TIPS only uses temporal information to determine the weight function (only the similarity weight). Once the weight function is determined, it becomes a weighted mean function and applies its convolution kernel one volume after another. It does not exploit the time continuity property of tissue time-concentration curves.

In our experiments, we re-implemented the 3D TIPS bilateral filter following their description and used it as a comparison. The kernel size chosen is 5×5. Furthermore, following the results of Mendrik's research, the standard deviation (μ_d) in their Gaus-

[10] 3D and 4D bilateral filters are the original bilateral filter mentioned in Section 3.1.2.2 with 3D and 4D (where the time dimension is treated the same as spatial dimension) kernels.

sian closeness function, determining which distance is considered close, was set to 4; the standard deviation (μ_ζ) in their TIPS function, which determines the maximum sum of squared difference up to which the time-intensity profiles are still considered similar, was set to 6.

3.2.4.2 Mean Filter

In the spatial mean filter (MEAN) used in the experiment, the value of each voxel is replaced with the mean value of its neighbours, including itself. This is a simple and intuitive method for smoothing images. Mean filtering can reduce the variation of intensity between adjacent voxels and thus reduce noise in images. In mean filtering, the first step is to group voxels by their coordinates. After that, the average value in a group, as the denoised voxel value, will be assigned to every voxel within the group. A mean filter can be used in both 2D and 3D kernels. In our experiment, a mean filter with a 2D kernel is used. In our experiment, the size of each group is 3×3. Each voxel in a kernel contributes the same value[11] to the denoised value. Furthermore, the denoise is performed volume by volume for all of the volumes in the time sequences.

3.2.4.3 Mean & GPR Filter

A method called MEAN & GPR is also introduced. Essentially, this method is a mean filter followed by a GPR denoising. It first uses a spatial mean filter to concentrate information from adjacent voxels and then uses Gaussian process regression to reduce the noise further. The mean filter used in this method also has a 3×3 kernel, the same as Section 3.2.4.2 above. Mean & GPR can deliver results which have the features of GPR denoising as well as the same resolution as MGPR.

Since the results of GPR and MGPR have different spatial resolutions, it is not straightforward to compare these two methods directly. The input of Mean & GPR filter is the same as MGPR, which combines the tissue time-concentration curves from a few adjacent voxels. The outputs of these two methods also have the same spatial resolutions. Therefore, the Mean & GPR filter can be used to help us in understanding the difference between GPR and MGPR.

[11] Every element in the weight matrix is 1/9.

Table 3.1: Patients Data

	Age	Gender	Time to Imaging (hh:mm)
Subject 1	74	Male	01:54
Subject 2	77	Male	05:30
Subject 3	78	Female	04:05
Subject 4	88	Female	01:50
Subject 5	83	Male	01:45
Subject 6	84	Female	02:40
Subject 7	82	Female	01:30
Subject 8	82	Male	01:30
Subject 9	51	Female	03:45
Subject 10	47	Female	04:40
Average	**75**	**M=4, F=6**	02:17

This table shows the age, gender and time to imaging from onset of the ten patients.

3.2.5 AIF Selection

In our experiment, global AIF technique is used. AIFs are determined using a free software application called Perfusion Mismatch Analyzer (PMA) [17]. It uses several criteria, such as peak value and arrival time, to calculate a score for each voxel and uses this score to find out the artery. The AIF is the average curve of the tissue time-concentration curves from the voxels with the highest scores.

3.2.6 Patients and Imaging Acquisition

The patient data used in our experiments is from the Multi-centre Acute Stroke Study (MASS), where patients have been recruited into a prospectively randomized trial. We used data from 10 patients (6 female and 4 male) with a median age of 80 years (maximum 88 years, minimum 47 years, average 75 years) recruited in a prospective observational study of patients with acute stroke within 6 hours of admission. All of the patients had a radiologically confirmed diagnosis of ischemic stroke. The average time to imaging from onset was 2 hours and 17 minutes (with a maximum of 5.5 and a minimum of 1.5 hours). Table 3.1 illustrates the details of patient information. All of these patients were confirmed to have ischemic stroke.

CT scans were obtained with *Siemens Somatom Sensation 16* scanners. Scans were performed with the patients head in plane with the orbitomeatal line, a slice thickness of 12mm, inter-slice gap of 1mm and tube current of 200 mAs, peak voltage of 80 KV and field of view (FOV) of 23cm. Due to limitations of the available equipment only

two slices[12] of perfusion data were available, however this is similar to the type of CT equipment in many hospitals currently. The images acquired are CT images with a data size of $512 \times 512 \times 2$ with 38 valid time intervals.

3.3 Experimental Results

3.3.1 Contrast to Noise Ratio

Contrast-to-noise ratio (CNR) is a quantitative parameter used to determine image quality. It is used to measure the noise level relative to the signal change. It is similar to the signal-to-noise ratio (SNR), but subtracts a bias before taking the ratio. The removal of bias is important when the bias is significant in an image. Since CT images have background intensities for all of the voxels, CNR is a better measurement for the image quality than SNR. CNR is widely used in the measurement of the noise in medical imaging (perfusion and non-perfusion, CT and MRI).

CNR is defined as one value for the whole image of a scan. It is defined as the reciprocal of the coefficient of variation [86].

$$\mathrm{CNR} = \frac{\mu - e}{\sigma} \tag{3.13}$$

where μ is the mean signal value, e is the bias and σ is the standard deviation of the noise. Since there is no contrast material injection in the first ten seconds of each scan, the oscillation in the first few seconds are pure noise. Therefore, the standard deviation of the first ten seconds is considered as the standard deviation of the noise. The bias, e, is set to a baseline value, which is the trimmed mean of the intensity values of the first ten seconds.

Furthermore, voxels outside the brain are excluded from the calculation of the CNR, thus the background area, with zero intensity values, does not affect the CNR.

3.3.2 Covariance Function Selection

The covariance function in Equation 3.9 is a squared exponential. The length scale l, the signal variance μ_f^2 and the noise variance μ_n^2 are required to be set in order to optimize the marginal likelihood. Among these three hyperparameters, the length scale is the most important one. Once the length scale is determined, the other two hyperparameters can be optimized based on l. If the length scale is too short, the output will

[12] There are only two validated slices, due to the reason stated in Section 3.1.4.

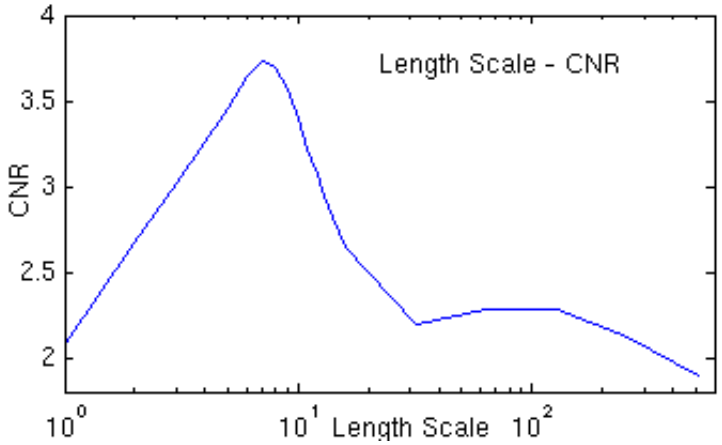


Figure 3.3: Length Scale Factor

This figure indicates the relationship between length scale l and CNR. Different length scales are applied to the same dataset in order to find out how CNR changes with the length scale.

be very unstable, as it will be over-fitted. Conversely, if the length scale is too long, the output will be over smoothed and this will lead to inaccurate results.

The selection of the length scale l is based on trial and error. All the subjects are processed with a different length scale and their average contrast-to-noise ratios are evaluated. Since the larger the CNR a length scale delivers the better the length scale is, the length scale is selected to reach the maximal CNR. Thus the peak at $l = 8$ in Figure 3.3 represents the turning point between over smoothing and insufficient smoothing, so the parameter l is set to 8 in order to achieve the largest CNR. The signal variance μ_f^2 is optimized to 1.1 and the noise variance μ_n^2 is optimized to 0.5.

To test the 9 possible length scale candidates[13], our perfusion imaging analysis program has to be executed 9 times, which takes about 100 seconds in total (details on the perfusion imaging analysis time can be found in Section 3.3.10).

The determination of length scale is based on one dataset, the determined result $l = 8$ is cross validated using two more datasets. It is then used as a fixed constant and applied to the rest of our datasets.

[13]The 9 tested length scales are:$2^0, 2^1, \ldots, 2^8$.

Table 3.2: Covariance Matrix for CT Images

1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00
0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00
0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00	0.00
0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00	0.00
0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00	0.00
0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01	0.00
0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01	0.01
0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01	0.01
0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02	0.01
0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03	0.02
0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04	0.03
0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05	0.04
0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07	0.05
0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09	0.07
0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12	0.09
0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15	0.12
0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19	0.15
0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24	0.19
0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29	0.24
0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36	0.29
0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43	0.36
0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50	0.43
0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58	0.50
0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67	0.58
0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75	0.67
0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83	0.75
0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90	0.83
0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97	0.90
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03	0.97
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07	1.03
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09	1.07
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10	1.09
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.03	0.04	0.05	0.07	0.09	0.12	0.15	0.19	0.24	0.29	0.36	0.43	0.50	0.58	0.67	0.75	0.83	0.90	0.97	1.03	1.07	1.09	1.10

This table shows the covariance matrix used in our method for the CT images used in the experiment. It is a symmetric matrix and the red elements are on the main diagonal.

3.3.2.1 Covariance Matrix for GPR

Table 3.2 shows the covariance matrix in Equation 3.4 for the CT images calculated from the given hyperparameters specified in Section 3.3.2. Element $k(i,j)$ at the i_{th} row (top to bottom), j_{th} column (left to right) indicates the covariance value between the i_{th} and j_{th} sampling time points[14]. In the table, it can be seen that the covariance values drop to approximately a half if two of the sampling time points are ten seconds apart from each other ($|i-j|=10$), compared to the covariance value for two stacked sampling time points. The covariance value keeps decreasing sharply to less than 10% of its maximal value when two time points are 17 seconds apart. The covariance matrix works in a manner similar to the weighted mean function in the weighted mean filter. Voxel a with twice the covariance value of voxel b to the target voxel c does not mean voxel a has a contribution twice as large as voxel b in the final value of voxel c. The covariance value can only demonstrate the trend that voxel a is more important than voxel b to voxel c, while a doubled value in mean function means a doubled contribution in the final value. A higher value in both the weight function and the covariance matrix indicates a larger contribution made by the referenced point to the final value of the target point.

3.3.3 Total Variation

Total variation [87] is introduced in order to measure the level of oscillation in the data. The average total variation, $V_{overall}$ is determined as follows:

$$V_{overall} = \frac{\sum_{v=1}^{N}\sum_{t=2}^{time}|f(t)-f(t-1)|}{\mathrm{N}} \tag{3.14}$$

where $time$ is the number of sampling time points, $f(t)$ represents the intensity value at a given time interval and N refers to the number of voxels in a volume.

A variant of Equation 3.14 is also introduced to measure the oscillation level in the baseline period. As there is no injection of contrast material in the first 10 seconds in each scan, the oscillation level in the baseline period is then calculated from the first ten time intervals and satisfies the following equation:

$$V_{baseline} = \frac{\sum_{v=1}^{N}\sum_{t=2}^{10}|f(t)-f(t-1)|}{\mathrm{N}} \tag{3.15}$$

[14] So $k(i,j)$ equals $k(j,i)$.

3.3.4 Standard Deviation

Another method to measure the oscillation level is to use the standard deviation. The following equation is used to evaluate the overall smoothness:

$$S_{overall} = \frac{\sum_{v=1}^{N} SD(v, time)}{\mathrm{N}} \tag{3.16}$$

where $SD(v, time)$ is the standard deviation of the intensity values among all the time points for the v_{th} voxel. Thus $S_{overall}$ stands for the average standard deviation of all the voxels (excluding background) in the brain. Another term, $S_{baseline}$, is defined to quantify the smoothness of the baseline period. $S_{baseline}$ is similar to $S_{overall}$ but the SD is calculated from the first ten time points instead of all the time points:

$$S_{baseline} = \frac{\sum_{v=1}^{N} SD(v, 10)}{\mathrm{N}} \tag{3.17}$$

where $SD(v, 10)$ is the standard deviation of the intensity values of the first ten time points for the v_{th} voxel.

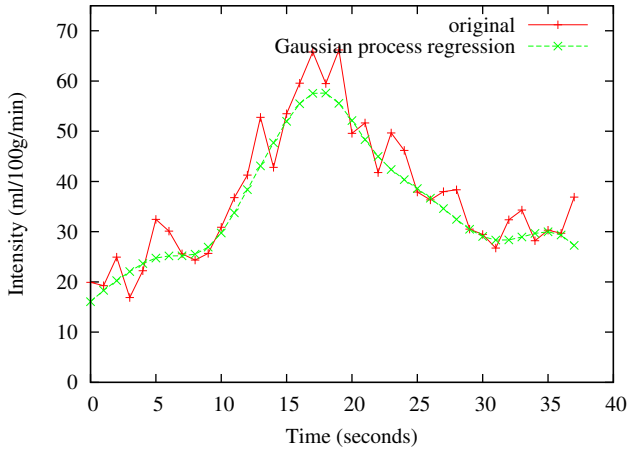
3.3.5 Gaussian Process Regression Denoising Results

This section focuses on the effectiveness of GPR noise reduction with respect to the parameters of individual voxels. Although hemodynamic quantities are the key terms evaluated from the individual voxel, tissue time-concentration curves are also very important since they are the data source, which determines the hemodynamic quantities, and may contain other information as well.

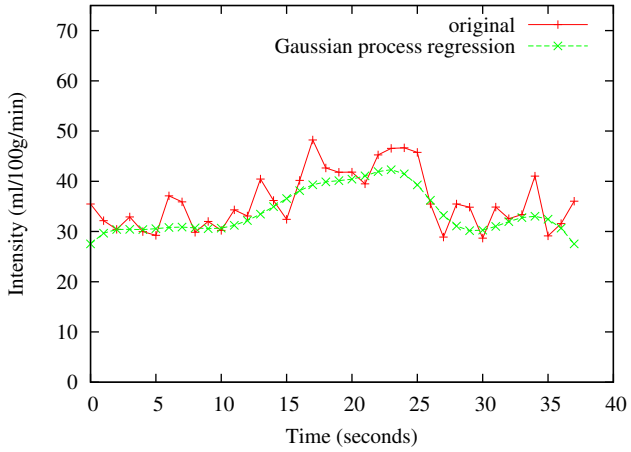
Typical results are shown in Figure 3.4. From this figure, it can be seen that GPR handles noise well for all of the voxels in grey matter, white matter and arteries. The three tissue time-concentration curves in the figure are obtained in voxels randomly selected from the given tissue types. Due to the fact that the tissue time-concentration curves are similar in the same tissue types in adjacent voxels, this figure is an example which reflects the improvement of our method in the three given tissue types[15]. In the arterial time series (Figure 3.4a) where the CNR is the highest, the application of GPR can perfectly fit the expected shape of the time-concentration curve. In the grey matter (Figure 3.4b) and the white matter (Figure 3.4c) where the CNRs are lower than in the arteries, GPR also produces significant improvement.

More specifically, there are three improvements when looking at the level of individual time series:

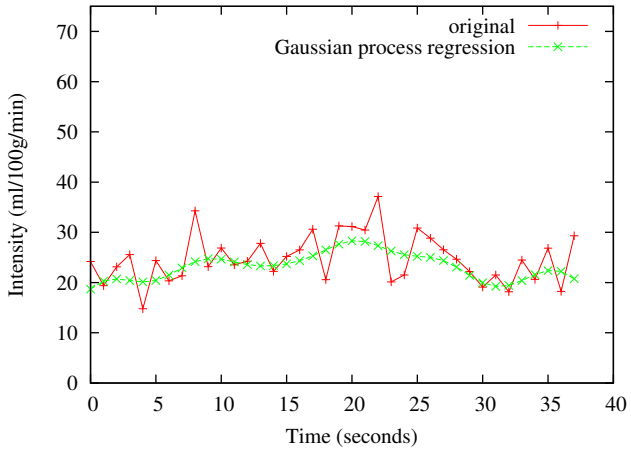
[15] Although it may not reflect the situation for some unusual voxels.



(a) Artery



(b) Grey Matter



(c) White Matter

Figure 3.4: Results of GPR

The figures show the raw and GPR processed time series of three randomly selected voxels corresponding to an artery, grey matter and white matter respectively. The y-axis represents the intensity value at each sampling time point.

Table 3.3: Raw versus GPR Denoised

	Raw	**GPR**	**GPR / Raw**
$V_{overall}$	182.2 ($\pm$33)	40.0 ($\pm$8.8)	22%
$V_{baseline}$	44.0 ($\pm$11)	9.6 ($\pm$3.9)	22%
$S_{overall}$	8.0 ($\pm$1.1)	6.5 ($\pm$1.0)	81%
$S_{baseline}$	5.0 ($\pm$1.7)	3.2 ($\pm$1.3)	64%

Comparative measurements of oscillation levels using the four parameters mentioned in Sections 3.3.3 and 3.3.4. The right most column indicates the remaining proportion of oscillations after denoising. So a small proportion value means a great improvement. The background area with straight zero intensity values is excluded from the calculation. The table represents the mean (with the standard deviation in brackets) value among ten subjects.

- First, considering the whole tissue time-concentration curve, the oscillations after denoising are much smaller than they are in the raw data. Table 3.3 illustrates the quantitative results. The mean (standard deviation) of total variation, calculated using Equation 3.14, among all the ten subjects before GPR is 182 ($\pm$33). The total variation falls dramatically to 40 ($\pm$8.8) with a reduction of 78% after applying GPR noise reduction. $S_{overall}$ also drops by 19%. However, it is problematic to use $V_{overall}$ and $S_{overall}$, since there is an increase in signal after the contrast arrival and a drop after it reaches the maximum concentration, so that both of the expected $V_{overall}$ and $S_{overall}$ values are non zero. For the raw data $V_{overall}$ is even larger than the maximal expected total variation[16]; GPR delivers results with $V_{overall}$ far less than that for the raw data.
- Second, GPR helps in determining parameters, such as Tmax[17] and bolus arrival time[18] from the output curves. As shown in Figures 3.4b and 3.4c, in the raw data, there are multiple peaks with similar peak values due to the noise and some of them are far away from the real peak (for example, the one at 8 seconds in the white matter). Even in the artery (Figure 3.4a), the oscillations can still lead to a bias in the Tmax. GPR ameliorates the situation and makes the determination more consistent.
- Third, considering the baseline period of tissue time-concentration curve, the baseline value calculated from the intensity values of the first few volumes[19] can be calculated more accurately due to the more stable onset of the tissue

[16] In an artery, the baseline value is usually 20-30 and the peak is usually 70-100. The amplitude in an artery is also the largest among all tissue types.
[17] The time corresponding to when the maximum contrast variation arrives.
[18] The time it takes for an injected bolus of contrast material to arrive at a given region of the brain.
[19] Usually the trimmed mean value of the first five, or a few more, seconds.

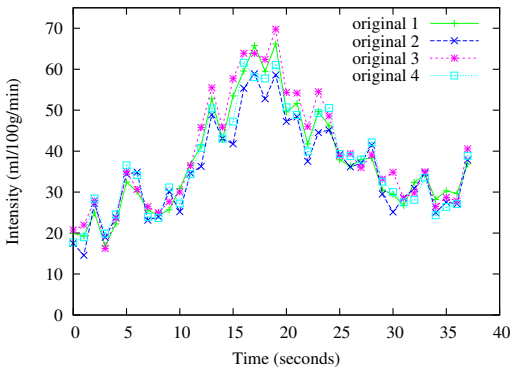
time-concentration curve. As the injection of contrast material occurs at approximately ten seconds after the scan starts, the signal in the beginning of the tissue time-concentration curve is expected to be constant. Thus the expected $V_{baseline}$ and $S_{baseline}$ values are both zero and any oscillation in the signal during the baseline period is pure noise. Quantitive measurements of the oscillations during the baseline period are the $V_{baseline}$ and $S_{baseline}$ obtained in Equations 3.15 and 3.17. GPR decreases the $V_{baseline}$ from 44 ($\pm$11) to 9.6 ($\pm$3.9). Meanwhile, the $S_{baseline}$ reduces from 5.0 ($\pm$1.7) to 3.2 ($\pm$1.3). The decrease by 78% and 36% for $V_{baseline}$ and $S_{baseline}$ respectively indicates the reduction in oscillations achieved by using GPR.

3.3.6 Multiple Observations Gaussian Process Regression Denoising

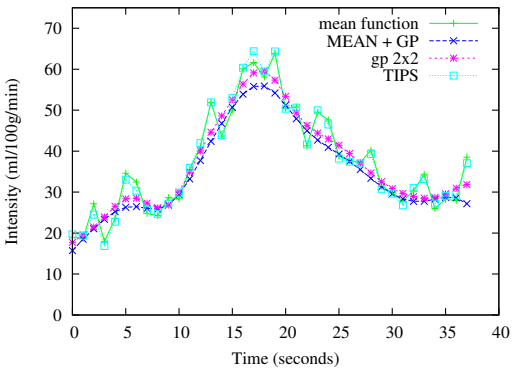
The result of our MGPR noise reduction method is presented in this chapter. To evaluate the effectiveness of MGPR, the results of four methods are presented for comparison. The first is the weighted mean filter (MEAN). The second is to add an extra GPR step after method one (MEAN & GPR). The third is MGPR as described in section 3.2.3. The fourth is the TIPS filter described in section 3.2.4.1. All of these four methods are based on or partly based on spatial decimation and have the same spatial resolution in their results.

Figure 3.5 shows how these four methods differ in their denoising efficiency in the context of 3×3 spatial decimation. The spatially weighted mean filter does not perform well for all of the three different tissue types, as it exhibits large oscillations. The TIPS filter denoised tissue time-concentration curves are still noisy with large oscillations. In the artery, because the tissue time-concentration curves in adjacent voxels are similar, the TIPS filter delivers results very similar to the mean filter. This situation also happens in the grey matter and white matter. Both MGPR and Mean & GPR deliver smooth results for the artery data; MGPR is the best for grey matter data, as it delivers a stable baseline, as well as retaining information about the signal's peak; in the white matter data, with the lowest CNR, MGPR still delivers a better baseline and peak value compared with the other two methods. Both the regression-based MGPR and the Mean & GPR methods deliver results which are smoother and contain fewer oscillations than results from the mean filter and the TIPS filter.

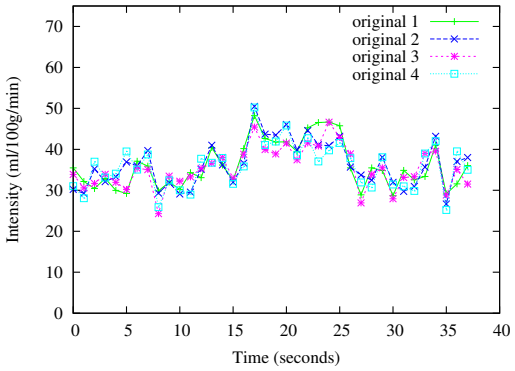
In conclusion, based on data from 10 subjects, the spatial mean filter does not



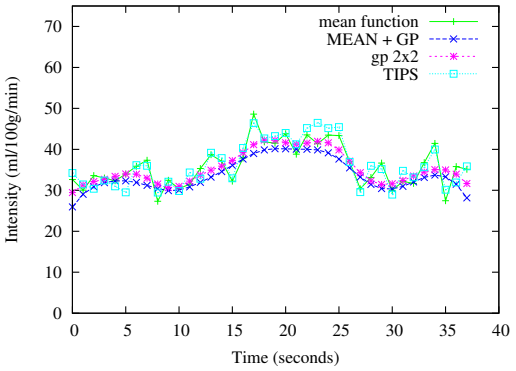
(a) Artery Raw



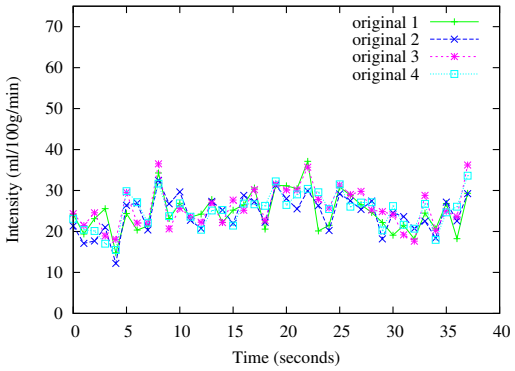
(b) Artery Denoised



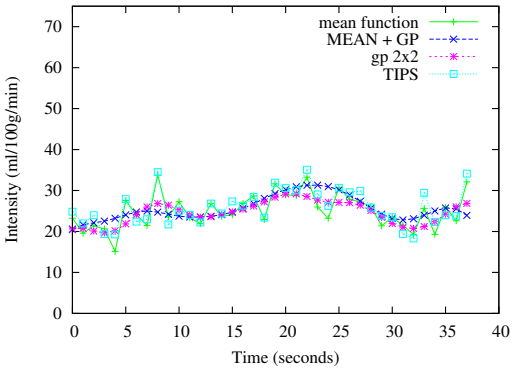
(c) Grey Matter Raw



(d) Grey Matter Denoised



(e) White Matter Raw



(f) White Matter Denoised

Figure 3.5: Comparison of Four Denoising Methods

The figures on the left show the raw data of the four (2×2) adjacent voxels. The ones on the right show the results processed by different denoising methods: spatially weighted MEAN filter (mean function), weighted mean filter plus regular Gaussian process regression (*MEAN + GP*), *MGPR* (gp 2×2) and TIPS bilateral filter (TIPS). Figures from top to bottom are for tissue types artery, grey matter and white matter. The y-axis represents the intensity value at each sampling time point.

Table 3.4: Contrast-to-Noise Ratio

	Raw	**TIPS**	**GPR**	**MEAN 3**	**GPR & MEAN 3**	**MGPR** 3×3
Subject 1	1.58	1.78	3.70	1.86	3.14	3.79
Subject 2	1.32	1.43	3.09	1.57	2.83	3.28
Subject 3	2.16	2.53	3.29	2.64	2.61	3.75
Subject 4	1.28	1.34	1.93	1.32	1.80	1.91
Subject 5	1.80	2.07	4.32	2.27	3.99	4.18
Subject 6	2.10	2.39	5.44	2.64	5.12	5.12
Subject 7	2.04	2.30	3.60	2.47	2.85	3.87
Subject 8	1.36	1.53	3.51	1.62	3.70	3.31
Subject 9	1.14	1.20	1.35	1.24	1.15	1.40
Subject 10	1.07	1.15	1.76	1.23	1.39	1.85

Comparison of the CNR of raw and denoised images for all of the ten subjects. The CNR displayed are the average CNR of the whole brain images (black background outside the brain is excluded). Column *Raw* is for raw data; column *TIPS* is for TIPS filter; column *GPR* is for Gaussian process regression; column *MEAN 3* is for $[3 \times 3]$ spatial weighted mean filter; column *GPR & MEAN 3* is to apply Gaussian process regression to $[3 \times 3]$ spatial weighted mean filter denoised data and column *MGPR 3* $\times$ *3* is for $[3 \times 3]$ MGPR.

Table 3.5: CNR Improvement

Optimized results	**Raw**	**TIPS**	**GPR**	**MEAN 3**	**GPR & MEAN 3**	**MGPR** 3×3
Mean	1	1.11	1.99	1.18	1.78	2.02
Standard Deviation	-	0.042	0.516	0.072	0.589	0.457

The average improvement of the CNR for the methods in Table 3.4 relative to the raw image. The CNR of each raw image is set to 1 for normalization.

perform as well as the other methods in this comparison; MGPR is better than MEAN & GPR because MGPR provides a higher CNR and does not lose detailed distribution information in the spatial mean filter step.

3.3.7 CNR Improvement

Tables 3.4 and 3.5 show that our basic GPR method gives a 99% higher CNR on average than the CNR for raw data. When using spatial decimation, MEAN, to obtain higher CNR, the approach using the spatially weighted mean filter only gives an 18% improvement for a 3×3 spatial decimation. Furthermore, the MEAN & GPR method shows its advantages in that it also improves the CNR by about 78%, which is much better than MEAN alone, but smaller than the CNR obtained using the GPR method. The TIPS bilateral filter method only gains 11% CNR improvement, which is much smaller than that obtained by our GPR.

A much better solution is to use the MGPR method which more than doubles the CNR. Of all the methods used in our comparison, regardless of the reduction in spatial resolution, MGPR delivers the highest CNR. Considering its effectiveness in CNR improvement and in tissue time-concentration curve improvement illustrated in Section 3.3.6, it is the best solution among all of the spatial decimation methods.

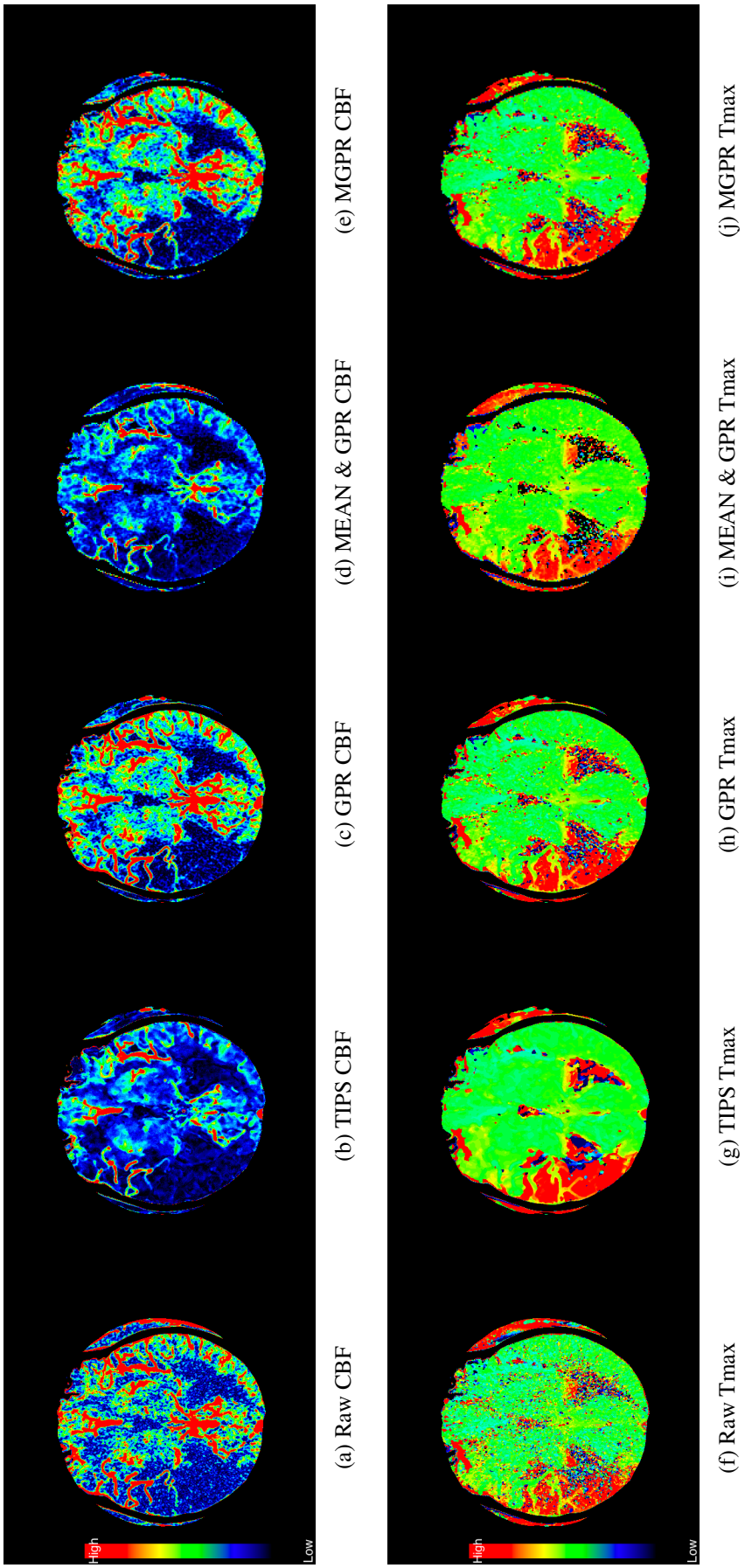
Furthermore, the CNR improvement can vary in different subjects. This may be caused by the difference in the amount of injected contrast agent and different injection speed. As illustrated in table 3.4, GPR improves the CNR of subjects 6 and 8 best with by 159% and 158%, respectively, while it only improves that of subject 9 by 18%.

3.3.8 Qualitative Results

CNR is not the only criterion to measure perfusion image quality. Another very important criterion is the quality of the parametric maps. Figure 3.6 shows the result of comparing the quality of CBF and Tmax maps when using GPR. The CBF value for each voxel was calculated using a truncated singular value decomposition (SVD) method [26, 27] with a threshold set to 0.15 [28][20]. In the raw CBF map (Figure 3.6a), the lesion area at the left bottom side may not be clear enough to be distinguished and the image is blurred. The TIPS bilateral filter (Figure 3.6b) produces less blurred images. GPR denoising gives the best result; the lesion area[21] can be noticed more easily and the image is much clearer (Figure 3.6c). GPR denoising also demonstrates its advantage in Tmax maps; the edges and detailed information can be identified better than in the raw data and in the TIPS bilateral filter denoised data, and consequently better than in the Gaussian, 3D bilateral or 4D bilateral filters. For the low-flow areas, the white matter and lesion area, GPR provided very good results.

[20] We have re-implemented their method based on their description.

[21] In ischemic stroke, lesion areas are the unexpected black (low-flow) areas inside the brain. A simple way to spot lesion areas is to compare with the corresponding area in the apposite brain hemisphere.

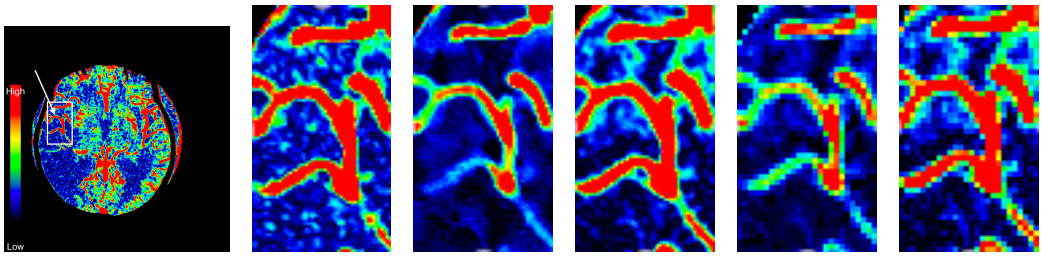


(a) Raw CBF (b) TIPS CBF (c) GPR CBF (d) MEAN & GPR CBF (e) MGPR CBF

(f) Raw Tmax (g) TIPS Tmax (h) GPR Tmax (i) MEAN & GPR Tmax (j) MGPR Tmax

Figure 3.6: CBF and Tmax Maps

Figures from left to right are raw, *TIPS*, *GPR*, *GPR* & *MEAN 3* and *MGPR* 3×3 denoised maps. The top row is for CBF maps and the bottom row is for Tmax maps. Results from the *MEAN 3* method are worse than the raw maps and are not presented here for comparison.



(a) Zoom In Area (b) Raw (c) TIPS (d) GPR (e) MEAN & GPR (f) MGPR

Figure 3.7: Full Size CBF

This figure contains full size CBF images of the selected rectangle shown in Figure 3.7a. Figures 3.7b to 3.7f are a 60×100 area of full sized 512×512 images.

Spatial decimation based GPR methods (MEAN & GPR and MGPR) provided relatively good solutions. However, in the full size (original size) CBF images in Figure 3.7, the results from the spatial decimation based method become a little blurred as their resolution is reduced by a factor of 9. TIPS and MEAN & GPR reduce the size of blood vessel in their processed CBF images. GPR (Figure 3.7d) keeps the right size of blood vessel and separates the vessel from other tissues well. GPR-processed CBF is also less blurred than the maps produced using MGPR (Figure 3.7f). GPR based methods (Figures 3.7d and 3.7f) seem to render the areas and boundaries of the original raw image (Figure 3.7b) with greater fidelity.

3.3.9 Experts' Opinions

We conducted a study in order to discover experts' preferences between hemodynamic quantity maps from raw images, our methods' denoised images and other methods' denoised images. Images were organized into groups and Figure 3.6 is an example of two groups of images in the questionnaire (one group for CBF images and one group for Tmax images). The questionnaire can be found in Appendix A (Questions 1 to 13). Experts were asked to rank images within each group in terms of quality (from 1 to 6, where 1 means the best one in the group and 6 means the worst one). As the questionnaire results are a ranking score rather than a weighted factor, a score of '1' can be only interpreted as better than a score of '2' but can not be considered as twice as good as score '2'. The reason for using ranking instead of scoring is that ranking shows the superiority of these six methods, so the preference level of these methods can be compared directly. The order of images in each group are disrupted to prevent inertia of thinking and any possible bias from participants. The name of the method

Table 3.6: Questionnaire Participants

	Specialty	**Years of Experience**
Participant 1	Neurologist	3
Participant 2	Medical Physicist, Neurologist, Geriatrician, Stroke physician, Radiologist	6
Participant 3	Geriatrician	17
Participant 4	Radiologist	4
Participant 5	Unspecified	6
Participant 6	Radiologist	Unspecified
Participant 7	Radiologist	25
Participant 8	Geriatrician	5
Participant 9	Radiologist	Unspecified
Participant 10	Stroke physician	12
Participant 11	Stroke physician	12
Participant 12	Neurologist	12

This table illustrates the specialty and years of experience for each participant.

Table 3.7: Questionnaire Results

	Raw	**GPR**	**MGPR** 3	**Mean 3**	**GPR & MEAN 3**	**TIPS**
Participant 1	5	3.75	3	4.25	3.17	1.83
Participant 2	4.75	2.42	2.58	4.42	4.17	2.67
Participant 3	4.42	2.75	2.5	4.25	2.83	2.5
Participant 4	2.83	2.83	2.5	3.67	3	2.67
Participant 5	3.25	2.25	2.58	3.08	3.83	2.5
Participant 6	3.67	2.5	1.83	4.17	3	2.33
Participant 7	5	2.75	2.92	4.08	4	2.25
Participant 8	4.75	3.75	3.33	4.17	3.17	1.83
Participant 9	4.67	3.75	3.08	4.67	3.17	1.67
Participant 10	5.25	3.17	3.25	5	2.67	1.67
Participant 11	4.58	3.5	3.58	4.08	3.5	1.75
Participant 12	4.88	3.88	3.05	4.29	3.29	1.72
Average Ranking Score	4.65	3.29	3.00	4.38	3.48	2.20
Standard Deviation	0.56	0.60	0.64	0.67	1.01	1.22

This table illustrates the average ranking scores which indicates experts' preferences.

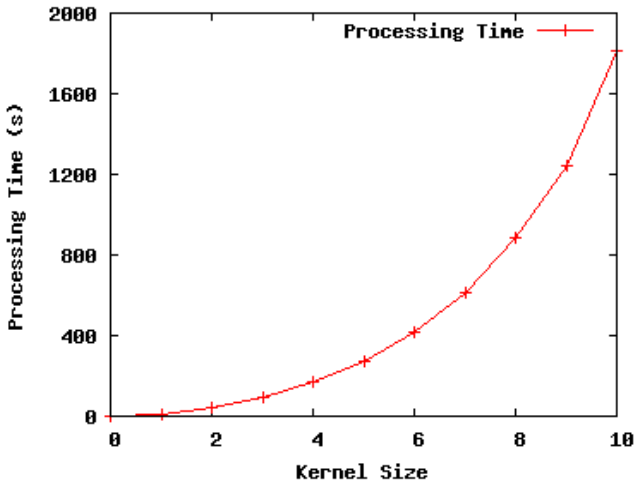


Figure 3.8: Processing Time

Kernel 1 represents the GPR method, kernel i ($i \neq 1$) means method MGPR with decimation factor $[i \times i]$.

that each image represents is also removed from the questionnaire.

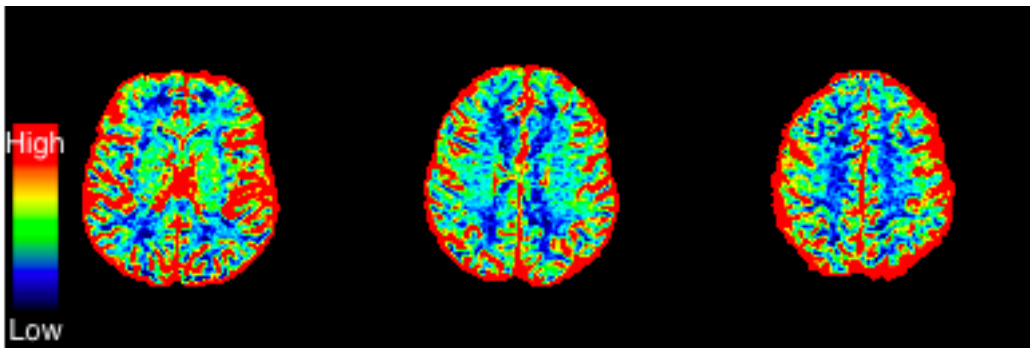
The results of the questionnaire were analyzed with the help of Dr. Francesca Chappell, a medical statistician in the Division of Clinical Neurosciences of the University of Edinburgh.

We used CBF and Tmax maps from 3 datasets, where each datasets have 2 slices, formed 12 groups (72 images). About 50 questionnaires were sent out to experts with diverse experience (such as neurologists, radiologists, stroke physicians and geriatricians) and we received 12 responses. Table 3.6 shows the specialty and experience for these 12 participants. Table 3.7 shows the results of the questionnaire. It indicates that GPR, MGPR and TIPS all received good scores. TIPS is the one with the best scores (2.20). However, it also has the largest standard deviation ($\pm$1.22), which means TIPS is not as stable as others. GPR is less case sensitive to cases than TIPS and delivers reasonably good score (3.29 $\pm$0.60). Furthermore, images from any of the GPR related methods, even GPR & MEAN, gained much higher scores than the raw images (4.65 $\pm$0.56).

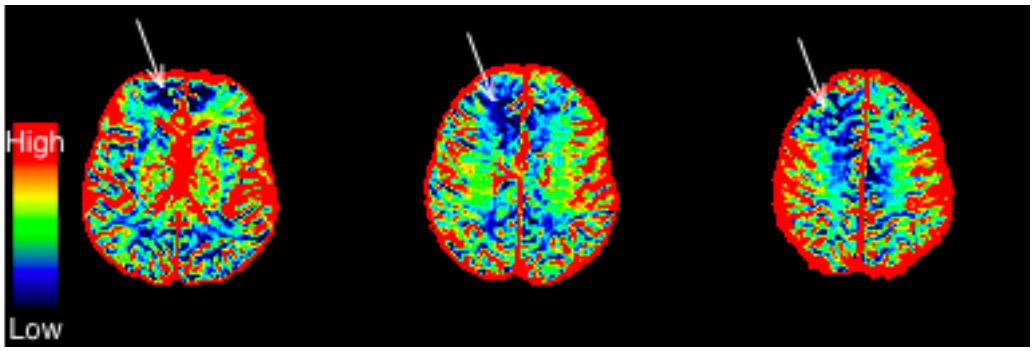
3.3.10 Processing Time

The running times of GPR and MGPR are evaluated on an $Intel(R)Xeon(R)$ CPU[22], with the details introduced in Section 2.3.1.1. Figure 3.8 illustrates the differences in processing time between our Gaussian process regression based methods for all the

[22]Only one core is used in the experiment, as both GPR and MGPR are implemented in serial.



(a) Raw



(b) GPR Denoised

Figure 3.9: Comparison of Raw and GPR denoised CBF for MRI

Visualisation of the impact of GPR on measuring CBF for MRI data. Each column is a pair and all three pairs are different slices from the same scan.

volumes in a scan[23]. Increasing the size of the voxel block dramatically increases the time needed for processing. GPR only takes 11.8 seconds, but it takes more than half an hour if we use $[10 \times 10]$ MGPR. The real processing time fits the theoretical computational complexity well:

$$Time(d_f) = Time(1) \times d_f^2 \tag{3.18}$$

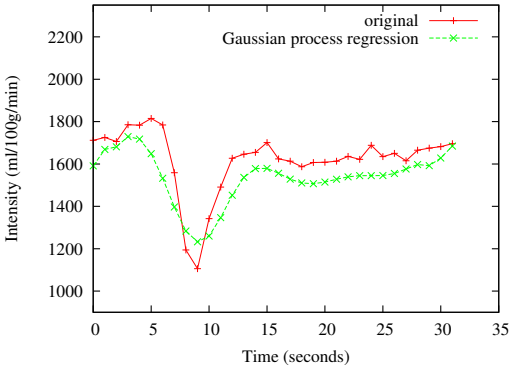
where d_f is decimation factor described in section 3.2.3.

In comparison, the processing time for 1×1 MEAN and 1×1 GPR & MEAN is less than 10 seconds and the processing time for TIPS is about 40 seconds.

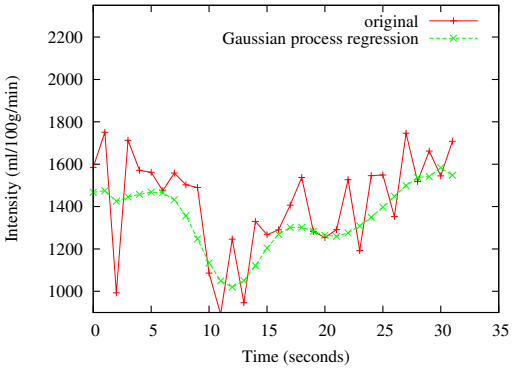
3.3.11 Magnetic Resonance Imaging Data

Magnetic Resonance Imaging (MRI) data have much higher CNR but a lower resolution than CT imaging data. As shown in Figure 3.10, using GPR to denoise tissue

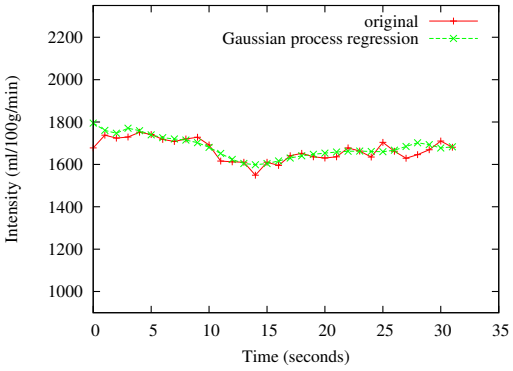
[23] At x-axis = 0, it means no noise reduction is applied, so that the denoising running time is 0.



(a) Artery



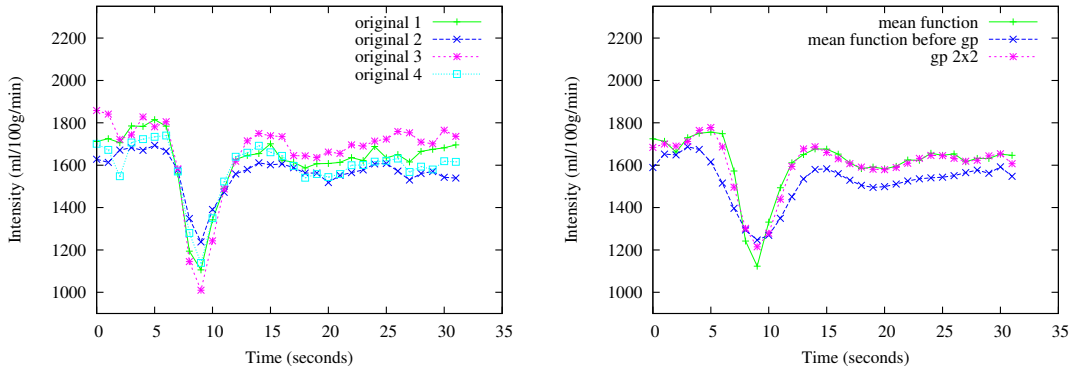
(b) Grey Matter



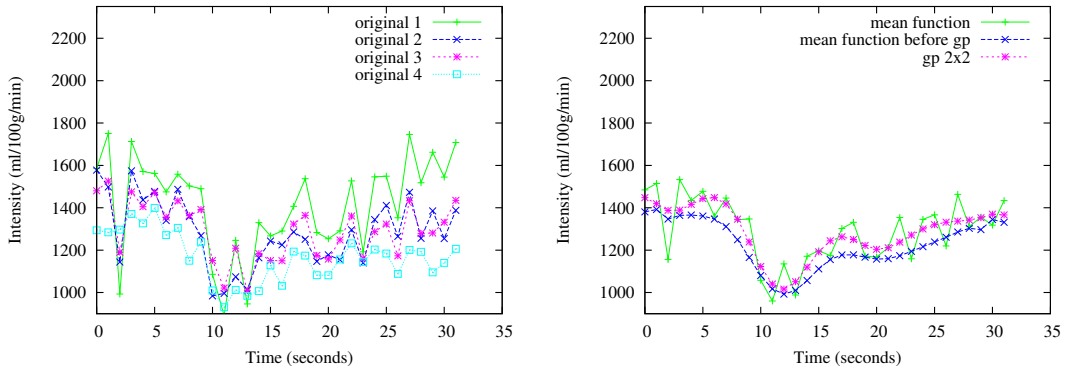
(c) White Matter

Figure 3.10: GPR for MRI Data

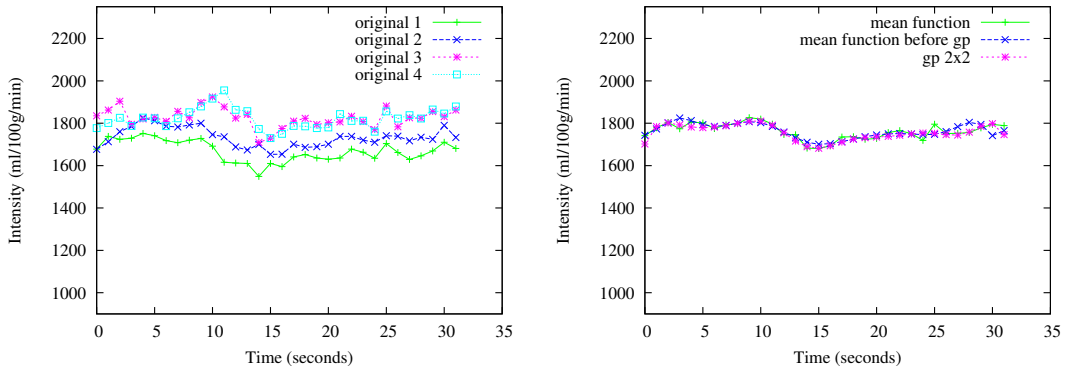
The same as Figure 3.4, figures show the raw and GPR processed MRI time series of three randomly selected voxels corresponds in an artery, grey matter and white matter respectively. The y-axis represents the intensity value at each sampling time point.



(a) Artery



(b) Grey Matter



(c) White Matter

Figure 3.11: MGPR for MRI Data

The figures on the left are the raw data for spatial decimation denoising of 2×2 kernels in different tissue types. The figures on the right are results using spatially weighted mean filter, GPR & MEAN and MGPR. The y-axis represents the intensity value on each sampling time point. This figure for MRI data corresponds to Figure 3.5 for CT data.

time-concentration curves produces a much smoother result than the original for all tissue types. GPR gives a reasonable baseline, peak value and Tmax. Figure 3.11 shows the results of spatial decimation methods. In artery, grey matter and white matter, both of the MEAN & GPR and MGPR methods provide results with fewer oscillations than the weighted mean filter approach.

Figure 3.9 illustrates an example of the improvement achieved by GPR in CBF maps. A lesion area can be identified at the anterior cerebral artery (top left hand side of the images) in the CBF maps delivered by GPR denoise, which is not so easily seen in the raw CBF image. GPR-processed CBF also provides more details in white matter and in the lesion area.

3.4 Conclusions

In this chapter, we present a noise reduction method using Gaussian process regression (GPR). It makes use of the temporal information in perfusion source images to reduce the noise. The smoothness level of its outputs is determined by hyperparameters, which are adjusted based on derivatives of the features of their inputs (CT perfusion images). We also developed a noise reduction method, called multiple observation Gaussian process regression (MGPR). It works as a combination of a spatial decimation filter and a temporal filter. MGPR reduces the resolution of the results but can benefit from being both a spatial decimation filter and a temporal filter.

Our methods are examined in four ways: by considering whole-brain summary statistics, by examining individual voxel's tissue time-concentration curves and hemodynamic maps, and by a usability study.

1. The results of GPR without spatial decimation show 99% improvement in CNR over raw data. Our spatial decimation based GPR also shows considerable improvement compared with the spatially weighed mean filter. For 3×3 spatial decimation, our MGPR and MEAN & GPR achieve 102% and 78% higher CNR than raw data respectively. This should be compared with the 18% improvement achievable by simply using a weighted mean filter. MGPR improves the CNR even a little more than GPR, regardless of the reduction in spatial resolution.

2. Considering the individual voxel, GPR denoised tissue time-concentration curves have much smaller oscillations than raw data, which helps to distinguish parameters, such as the baseline value and Tmax, more easily and with better accuracy.

3. For hemodynamic parametric maps, our GPR methods provide a much better solution with clearer edges and more detailed information achieved by the than other approaches we studied. These results show that Gaussian process regression based methods handle noise better than comparable techniques used.

4. In the questionnaire using experts' marks, results without denoising (raw images) received a very low ranking score of 4.65 ($\pm$0.56). TIPS bilateral filter received the best score of 2.20 ($\pm$1.22). Meanwhile, GPR and MGPR are also preferred with scores of 3.29 ($\pm$0.60) and 3.00 ($\pm$0.64), respectively.

All of these support the hypothesis that GPR is effective.

The running time for GPR is 11.8 seconds, while 3×3 MGPR costs about 100 seconds. Our methods, by improving the quality of output hemodynamic maps in reasonable time, have potential use for clinical diagnosis and brain research.

Chapter 4

Automatic Lesion Area Detection

4.1 Introduction

In this chapter, we propose a correlation coefficient tests based approach to spot the lesion area automatically. This approach makes use of perfusion source images directly. Correlation-coefficient tests are used in the approach to measure the similarity between expected tissue time-concentration curve and unknown time-concentration curves. This information is then used to differentiate penumbra and dead tissues from healthy tissues. The goal of the segmentation is to fully utilize information in the perfusion source images. This approach is designed to handle CT perfusion images, but it can also be used to detect lesion areas in MR perfusion images.

4.1.1 Motivation

CT perfusion imaging is widely used to calculate brain hemodynamic quantities such as cerebral blood flow, cerebral blood volume and mean transit time that help the diagnosis of acute stroke. Since brain perfusion source images contain more information than hemodynamic maps, a good utilization of the source images can lead to better understanding of scan results. Our method makes use of the perfusion source images and identifies lesion areas automatically using a correlation test. It helps to spot the lesion areas as well as to determine their boundaries. It can be used as additional information to help with the diagnosis of acute stroke.

4.1.2 The Use of Perfusion Source Images

As perfusion source images contain more information than cerebral hemodynamic parametric maps, good utilisation of the source images can lead to better understanding than hemodynamic maps alone. Wang et al.[88, 89] stated that the value of the CT perfusion source imaging has not been fully investigated in traditional perfusion methods and showed the potential in using CT perfusion source imaging information rather than CBF and CBV maps to help clinical diagnosis. They also show that the arterial phase and venous phase CT perfusion source imaging mismatch model could possibly be applied to ischemic regions in the acute stage of stroke to determine penumbra and infarct core.

Pepper *et al.*[90] have also illustrated how CT perfusion source images can be used in identifying acute ischemic change. Their findings suggest that CT perfusion source images may help to screen potential thrombolysis candidates.

4.1.3 Pattern Recognition and Correlation Analysis

Pattern recognition is a machine learning technique. It can be used as a classification tool, which attempts to assign each input value to one of a given set of classes. Compared with pattern matching, another category of algorithm, pattern recognition, aims to use a probable match and categorise all possible inputs instead of looking for exact matches in the input.

Correlation analysis can be used in pattern recognition since it can measure the level of similarity between the given set of classes and the inputs. In statistics, correlation refers to any of a broad class of statistical relationships involving dependence. In our method, a correlation coefficient test is used to measure the similarity between the time-concentration curves in healthy tissues and measured tissues, whose result can then be used to determine whether or not the measured tissues are abnormal. Correlation coefficients focus on the shape of tissue time-concentration curves, rather than in the intensity values. This feature is able to minimize the harm and the misunderstanding in diagnosis which is caused by noise and the differences in intensity values between different scans.

In perfusion imaging, different datasets have different lengths of waiting time before the injection of a contrast agent and different intensities of signal values. Therefore, different datasets need to be trained separately, since the features of their signal curves, for both of the healthy and abnormal tissue-categories, can vary. In order to

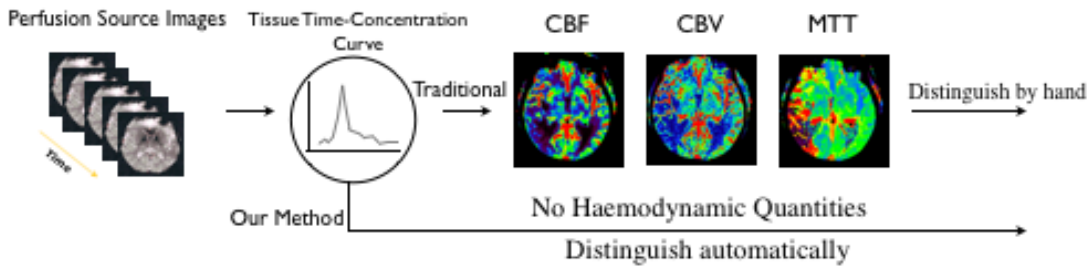


Figure 4.1: Workflow of Our Method

keep the training simple, unsupervised learning was used in the method, which means the training datasets have not been labeled by hand.

4.2 Materials and Methods

In statistics, correlation refers to any of a broad class of statistical relationships involving dependence. In our method, the key idea is to use correlation coefficient tests to measure the similarity of the tissue time-concentration curves between healthy reference tissues and target tissues (the tissues measured from a scan without knowledge of their states). Since the tissue time-concentration curves in artery, grey matter and white matter are expected to have the same shape (with different amplitudes), our method do not distinguish different tissue types and treat them accordingly. Thus, tissue time-concentration curves from all the voxels in a scan are compared to a single (scan-based) reference curve. The result can then be used to determine whether or not the measured tissues are abnormal. Measured tissues that have similar tissue time-concentration curves to the healthy tissue are considered as healthy. On the other hand, those tissues which have low correlation to the healthy reference tissue will be marked as abnormal. Furthermore, the level of correlation can also reflect the severity of damage.

Correlation coefficients focus on the pattern of the tissue time-concentration curve, rather than intensity values. Besides, correlation coefficients treat the tissue time-concentration curves as an entirety, instead of as separate intensity values. These features are able to reduce the possible harm and misunderstanding in diagnosis, which is caused by noise and differences in intensity.

ALGORITHM 3 - AUTOMATIC LESION AREA DETECTION

```
1  For count ← 0 to sizeof(image) − 1 {
2      image[count] ← Preprocessing(image[count])
3      image[count] ← remove_head_end(image[count], 2)
4      }
5
6  ref ← generate_ref()
7
8  For count ← 0 to sizeof(image) − 1 {
9      ρ or r ← correlation_test(image[count], ref)
10     color[count] ← translate(ρ or r)
11     bitmap[count] ← Student's t test(ρ or r)
12     }
13
14 return color[ ] and bitmap[ ]
```

4.2.1 Algorithm for Automatic Lesion Area Detection

As illustrated in Figure 4.1, in traditional dynamic contrast-enhanced susceptibility-weighted perfusion imaging methods, perfusion source images are used to generate maps of hemodynamic quantities, such as CBF, CBV, MTT and Tmax. These maps will then be used by clinical scientists to manually identify lesion areas. Compared to traditional methods, our method uses the perfusion source images to identify lesion areas directly. Algorithm-3 explains how the automatic lesion area segmentation method works.

Lines 1 to 4 correspond to the image preprocessing. The preprocessing focuses on reducing the noise in the source images. Specifically, Gaussian process regression (Chapter 3) is used to achieve voxel-by-voxel denoising. Then, first and last two voxels are removed from the tissue time-concentration curve for reasons given below in Section 4.2.2.

The second step is to define a reference which indicates the ideal shape and pattern of healthy tissue time-concentration curve (Line 6). The details of reference curve selection will be stated in Section 4.3.2.

Lines 8 to 12 correspond to the lesion area segmentation step. For each voxel of

the image, a ρ or *r* value (correlation value) between the voxel itself and the reference is calculated using either Pearson's correlation coefficient [91] or Spearman's rank correlation coefficient [92]. This ρ or *r* value is then interpreted into a color value or a bit value using Student's t-test. At last, graphical output is generated (Line 14).

4.2.2 Image Preprocessing

For the reasons stated in Section 3.1.1, CT perfusion images suffer a high level of noise.

The noise in perfusion source images badly influences the correlation test results since it makes the pattern of tissue time-intensity curves hard to recognize. In our method, Gaussian process regression (mentioned in Chapter 3) is used to preprocess the input images.

The reason for using Gaussian process regression, a voxel intensity curve-based denoising method, instead of any of the other image based noise reduction methods, is that the correlation tests are also based on the voxel intensity curve and not on the image.

In any voxel, as the start and the end of the tissue time-concentration curve have a low intensity value, they are very sensitive to noise and thus are usually removed from the calculation. In the preprocessing, the first and last two time points are removed.

4.2.3 Correlation Coefficient

The correlation coefficient is a measurement of linear dependence between two variables or two sets of data. A familiar example is to use correlation coefficient to evaluate the dependence between the demand for a product and its price. In our method, we consider the concentration curves from healthy tissue and measured tissue as two variables. The correlation test then measures the level of similarity between these two variables.

As we focus on the shape of the time-concentration curve and the relationship between different time points of each voxel rather than the absolute intensity values, correlation tests are good at handling our problem. Pearson Product-Moment Correlation Coefficient and Spearman's rank correlation coefficient are two of the most popular types of correlation coefficients. Both of these correlation coefficients were considered in our experiments.

4.2.3.1 Pearson Product-Moment Correlation Coefficient

Pearson Product-Moment Correlation Coefficient, which is also called Pearson's correlation coefficient, is defined as the covariance of two groups of numbers divided by the product of their standard deviations:

$$r_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \cdot \sigma_Y} \tag{4.1}$$

where r is the Pearson's correlation coefficient, $\text{cov}(X,Y)$ is the covariance of group X and Y.

Covariance is a measure of how much two variables change together and can be defined as:

$$\text{cov}(X,Y) = \text{E}[(X - \text{E}[X])(Y - \text{E}[Y])] \tag{4.2}$$

where $E[X]$ is the expected value of X, as all of the x_i in X are equally likely in our case, the covariance function and standard deviations can be simplified to:

$$\text{cov}(X,Y) = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{n} \tag{4.3}$$

$$\sigma_X \cdot \sigma_Y = \frac{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}}{n} \tag{4.4}$$

where n is the number of time points, x_i is the value of i_{th} element in X and $\bar{x}$ is the mean of group X.

Therefore, Equation (4.1) can be written as:

$$r = \frac{\text{cov}(X,Y)}{\sigma_X \cdot \sigma_Y} = \frac{\sum(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum(x_i - \bar{x})^2 \sum(y_i - \bar{y})^2}} \tag{4.5}$$

4.2.3.2 Spearman's Rank Correlation Coefficient

Spearman's rank correlation coefficient, which is also called the grade correlation or rank correlation, is similar to Pearson's correlation coefficient. The difference between these two correlation coefficients is that Spearman's rank correlation coefficient uses rank scores of elements in each group while Pearson's correlation coefficient uses their value. The Spearman's correlation coefficient is computed from these:

$$\rho = \frac{\sum_i (x_i' - \bar{x'})(y_i' - \bar{y'})}{\sqrt{\sum_i (x_i' - \bar{x'})^2 \sum_i (y_i' - \bar{y'})^2}} \tag{4.6}$$

where ρ is the Spearman's rank correlation coefficient, x_i' is the rank score for i_{th} element in group X, for example, if x_i is the k_{th} smallest value in the group, then $(x_i' = k)$. $\bar{x'}$ is the mean of x_i'. Since x_i' is the mean of the rank, it equals $n/2$, where n is the number of time points.

4.2.3.3 Interpretation of Correlation Values

For both the Pearson's correlation coefficient and the Spearman's rank correlation coefficient, the (ρ or r) value ranges from -1 to 1. A correlation coefficient value of 1 means that the target group is perfectly correlated with the reference group; a correlation coefficient of -1 means these two groups are perfectly inversely correlated; a correlation coefficient of 0 means that the two groups are not related. Generally, the larger the absolute value is, the closer the two groups are. In our case, measured tissue should not inversely correlate with the reference and hence both of the ρ or r values should be positive.

After the determination of correlation coefficient ρ or r value, there are two ways to determine whether or not the measured group is healthy.

The first one is to display the ρ or r values directly in a color map. Since correlation results can be any value in the range of zero to one, they are mapped to colors from cold to warm. More specifically, the mapped colors range from black (the lowest), blue, green, yellow, to red (the highest)[1]. The advantage of this method is that it reflects the probability of each voxel being abnormal. Red color indicates healthy tissue, the rest of the colors designate abnormal tissues with different levels of damages.

The second one is to state the null hypothesis: the target group (tissue in a voxel) and the reference group (tissue in a voxel) have the same shapes and patterns in their tissue time-concentration curves, and then use Student's t-test [93, 94] to validate this null hypothesis. The distribution of correlation coefficients follows Student's t-distribution with degrees of freedom $N-2$,

$$t = r\sqrt{\frac{N-2}{1-r^2}} \quad or \quad t = \rho\sqrt{\frac{N-2}{1-\rho^2}} \tag{4.7}$$

where N is the number of sampling time points, and ρ or r is the correlation coefficient value calculated using the Pearson's correlation coefficient or the Spearman's correlation coefficient, respectively. As correlation-coefficient values are expected to be positive using our method, a one-tailed t-test will be used. Once the t-value is determined, it can be found using a table of values from the one-tailed Student's t-distribution. If the p-value is below the threshold for selected statistical significance (in our case, 0.05 and 0.01), then the null hypothesis is rejected and we consider the target tissue as abnormal tissue. Otherwise if the Student's t-test fails to reject the null hypothesis,

[1] So the lesion areas (ischemic stroke) in our correlation results appear in dark value, the same as the lesion areas in CBF map. Our color encoding also keeps the continuity of color.

we define the measured tissue as healthy tissue. For Student's t-tests, as the results can only be accept or reject, only two colors will exist in the non-background area of the results: red, meaning acceptance (normal tissue), and white, meaning rejection (abnormal tissue)[2].

4.3 Results

4.3.1 Patients and Imaging Acquisition

The patient data came from Multi-Centre Acute Stroke Imaging Study funded by the Translational Medicine Research Collaboration (TMRC).

The details of the 10 patients' data used in this chapter are the same as those for denoising investigation - see Section 3.2.6.

The data and images used in the following section as an example correspond to one of the ten patients and for that patient the time to imaging from onset was 1 hour and 54 minutes.

4.3.2 Reference Selection

Since the patterns in the source images vary, especially as the length of the period before the contrast agent injection varies, in different scans, the healthy reference curve has to be determined separately for each scan. In the experiments, the artery input function (AIF) is used as the reference. On the one hand, the reason is that the artery has the largest contrast-to-noise ratio among all of the tissue types, which makes the pattern of tissue time-concentration curve the most clear. On the other hand, traditional methods, which use deconvolution to generate hemodynamic quantities, also require an AIF as input. This choice also benefits from existing AIF-selection methods. Furthermore, since correlation coefficient between healthy artery and grey (white) matter tissues are large (> 0.9), only one reference curve are used for all tissue types.

As in the previous experiments, in this experiments AIFs are determined using the same AIF technique mentioned in Section 3.2.5. The acquired AIF is then denoised using the GPR method mentioned in Chapter 3 in order to smooth the AIF and enhance the pattern.

[2] In this case, white, instead of black, is used to identify lesion areas in order to make the lesion areas more conspicuous.

4.3.3 Validation Parameters

In our datasets, the overall number of time intervals is 40 per perfusion dataset and as the first and last two time intervals are removed to reduce the noise impact (Section 4.2.2), there are 36 time intervals left. So the degrees of freedom, $N-2$, are 34 in the one-tailed Student's t-test in our case. From the Student's t-distribution table, the t values for 0.05 and 0.01 statistical significance level are 1.691 and 2.441, respectively.

According to Equation 4.7, the formula to calculate the ρ or r value is:

$$r \; or \; \rho = \sqrt{\frac{t^2}{N-2+t^2}} \tag{4.8}$$

So, to reject the null hypothesis at 0.05 statistical significance level the correlation coefficient ρ or r value has to be ≥ 0.279. The threshold of 0.01 statistical significance level is equivalent to a $\geq$0.386 ρ or r value.

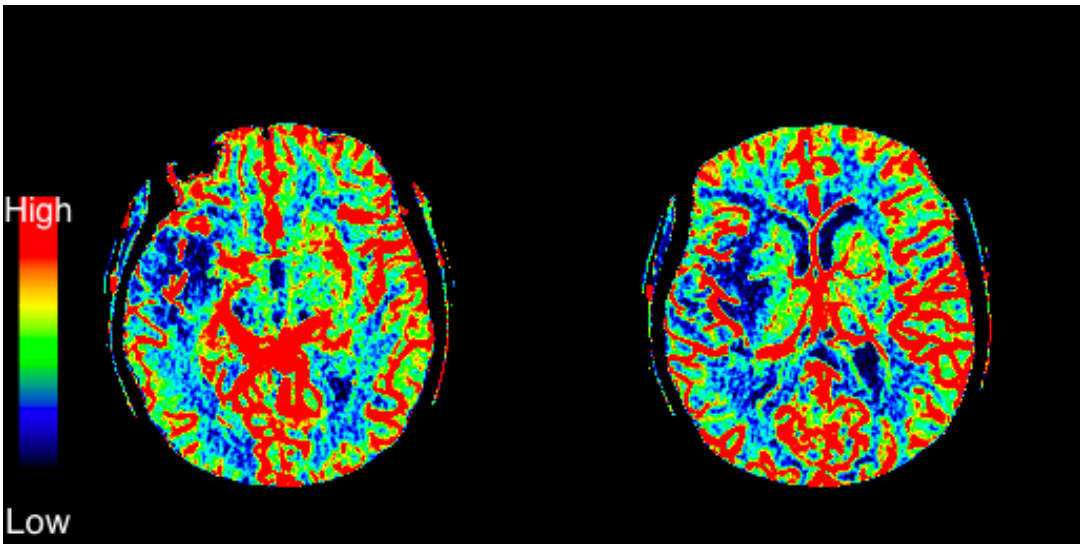
4.3.4 Hemodynamic Maps

Figure 4.2 shows hemodynamic maps of one single scan using a traditional analysis method, a truncated singular value decomposition (SVD) method [26, 27]. The threshold for the truncated SVD were set to 0.15 [28]. Our GPR noise reduction method mentioned in Chapter 3 is also used in preprocessing and contributes to the hemodynamic maps.

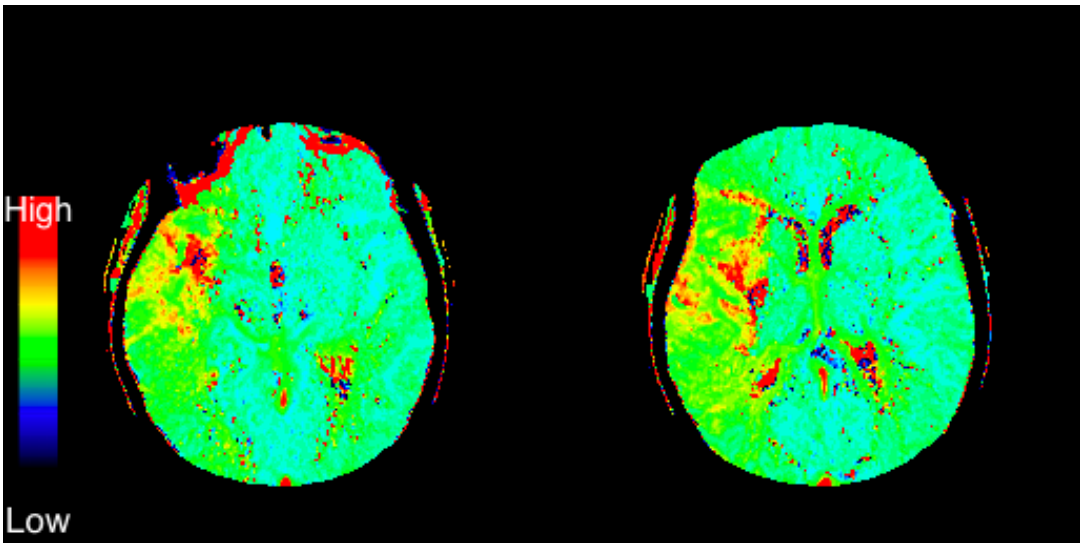
From the CBF map for slice 1 (Figure 4.2a), there is an area with unexpectly low blood flow at the middle left area. However, this phenomenon is less clear in its adjacent slice (Figure 4.2b). In both of the Tmax maps (Figures 4.2c amd 4.2d), it can be seen that the Tmax values are much larger (red and yellow) than the expected value in the contralateral hemisphere. Tmax maps also suggest that a larger part of the left-brain tissue (green) may be impacted by the stroke.

4.3.5 Validation of Results for Individual Voxels

We applied our method to randomly selected voxels of different tissue types. This experiment is repeated many times with different voxels selected, which deliver similar results. Figures 4.3 and 4.4 compare tissue time-concentration curves with the healthy tissue reference curve for different tissue types, without and with preprocessing (Section 4.2.2 and Chapter 3), respectively. Since the preprocessing is very important (the



(a) Slice 1 - CBF (b) Slice 2 - CBF



(c) Slice 1 - Tmax (d) Slice 2 - Tmax

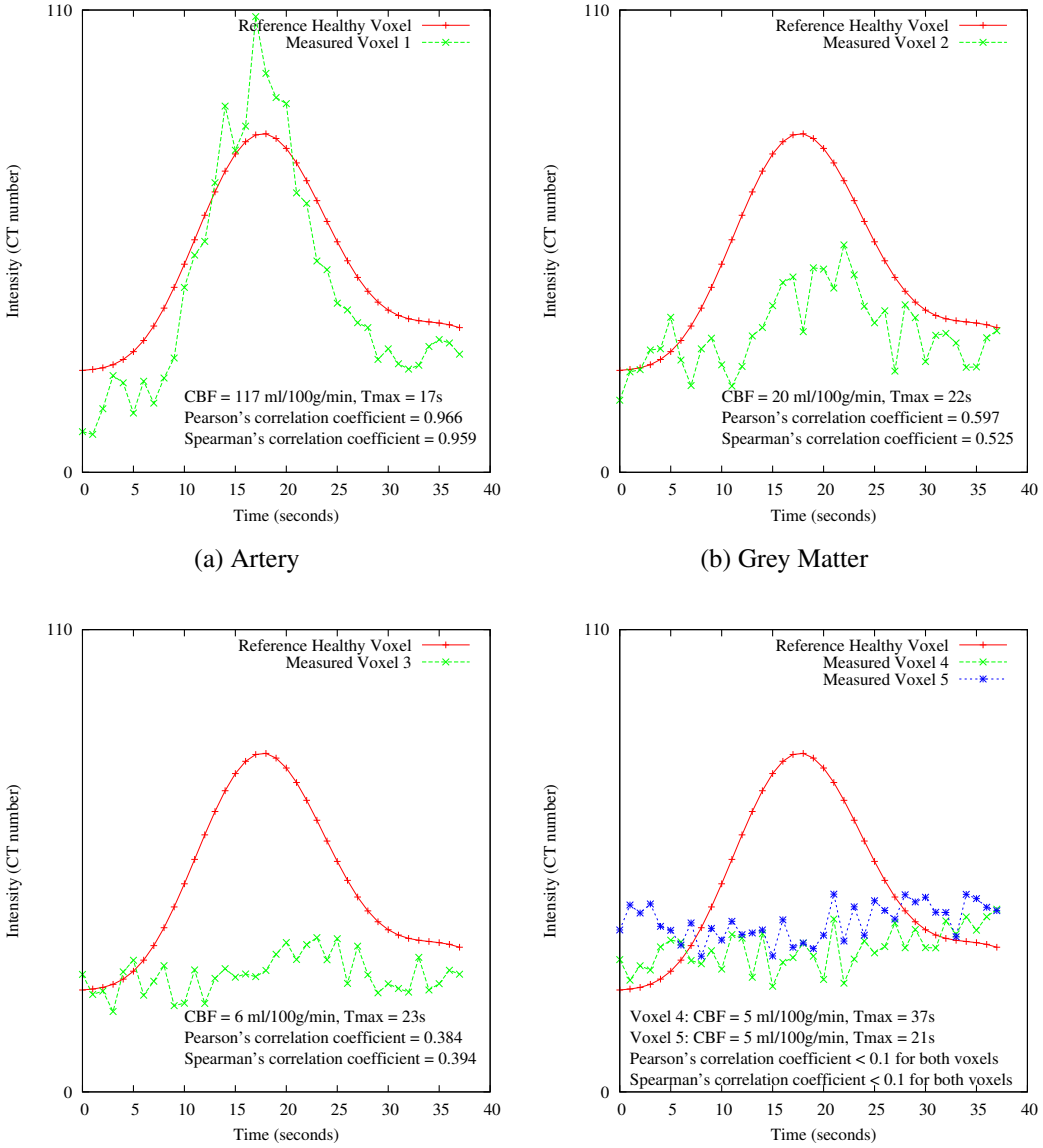
Figure 4.2: Reference CBF and Tmax (using Traditional Method)

These figures are CBF and Tmax maps used as references. The four figures are hemodynamic quantity maps from one patient. The top two are CBF maps for two adjacent slices and the bottom two are Tmax maps.

reasons are stated in Section 4.3.6), all of these correlation coefficient ρ or r values below are calculated using preprocessing to obtain noise reduced data.

The curve measured in an artery (Figure 4.4a) is the most similar to the reference curve. The r and ρ values using Pearson's correlation coefficient and Spearman's rank correlation coefficient, between the denoised artery curve and the reference curve, are 0.988 and 0.984[3], respectively. This is because the reference curve is generated based

[3]The r and ρ values are also large in the raw images (Figure 4.3a).

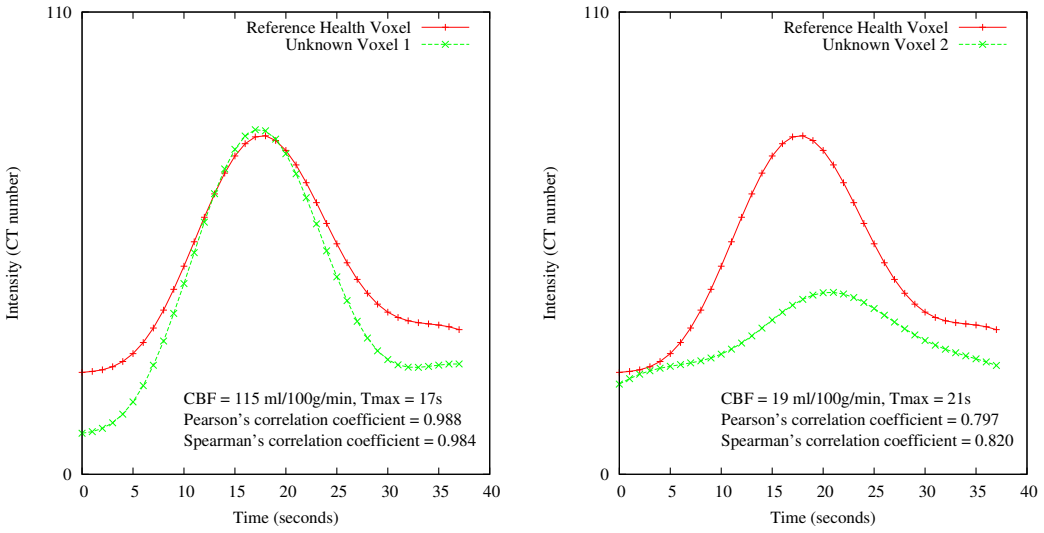


(a) Artery (b) Grey Matter

(c) White Matter (d) Lesion

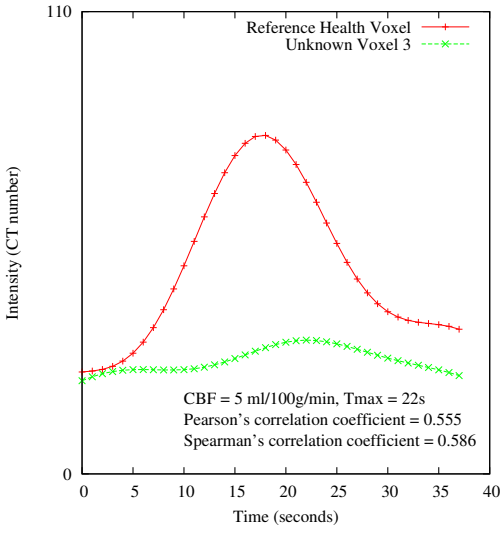
Figure 4.3: Reference Vs. Different Tissue Types (Raw Curves)

These figures show the raw tissue time-concentration curves in different tissue types. They are randomly selected from the given tissue types.

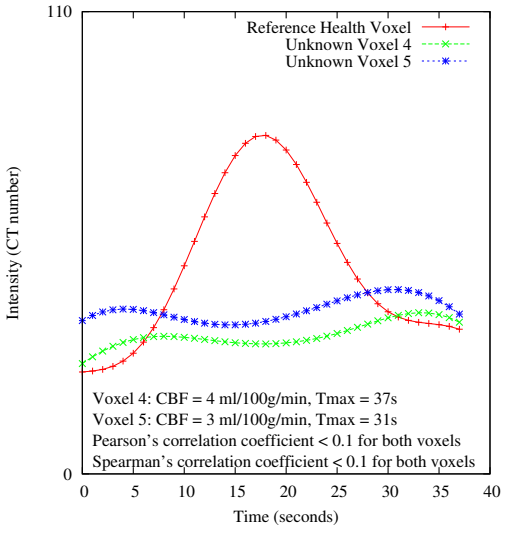


(a) Artery

(b) Grey Matter



(c) White Matter



(d) Lesion

Figure 4.4: Reference Vs. Different Tissue Types (Denoised Curves)

These figures show the denoised (using preprocessing) tissue time-concentration curves in different tissue types. They correspond to the same voxels as the ones in Figure 4.3.

on artery tissue, so that ρ or r values close to 1 are expected.

The ρ/r value using Pearson's/Spearman's correlation coefficient is 0.797/0.820 for grey matter (Figure 4.4b) and is 0.555/0.586 for white matter (Figure 4.4c). The r and ρ values in both the healthy grey and white matter voxels are large enough to reject the given null hypothesis at 0.01 significance level.

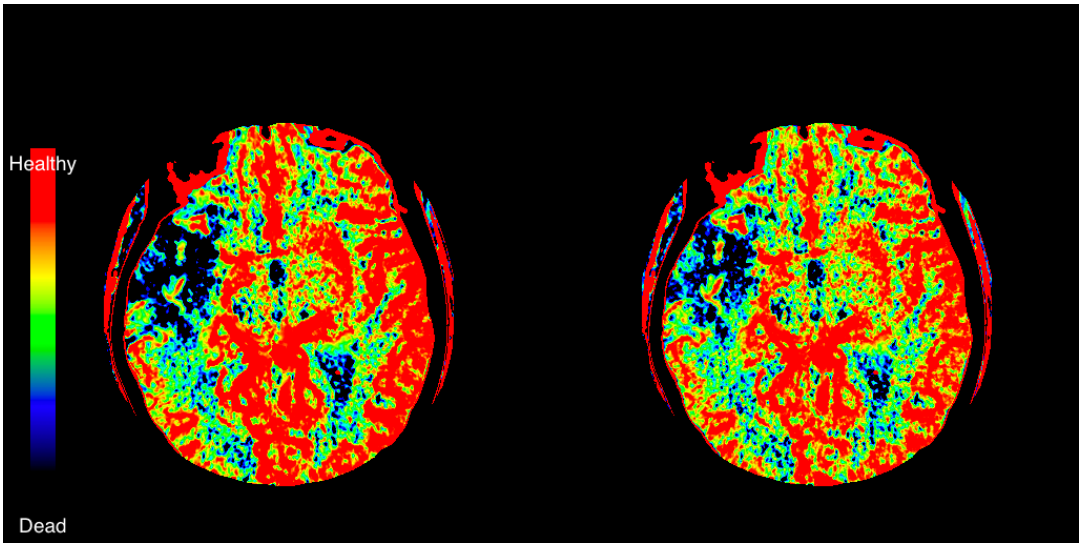
Measured voxels 4 and 5 (Figure 4.4d) are from a lesion area. Both of them have a low ρ or r value (< 0.1), which means these two measured voxels have curves that are significantly different when compared with the reference curves. As a result, these two voxels can be identified as abnormal tissue.

Since the intensity values are low (due to the nature of white matter) and fluctuate (due to noise) in both white matter and lesion areas, without correlation validation, it will be difficult to distinguish a lesion area from white matter by reading their tissue time-concentration curves. However, the measured voxel 3 in white matter has a ρ or r value larger than 0.5 while the ρ or r values in a lesion area is smaller than 0.1. Hence, correlation validation can help us to understand tissue status that cannot be easily found directly at the level of an individual voxel.

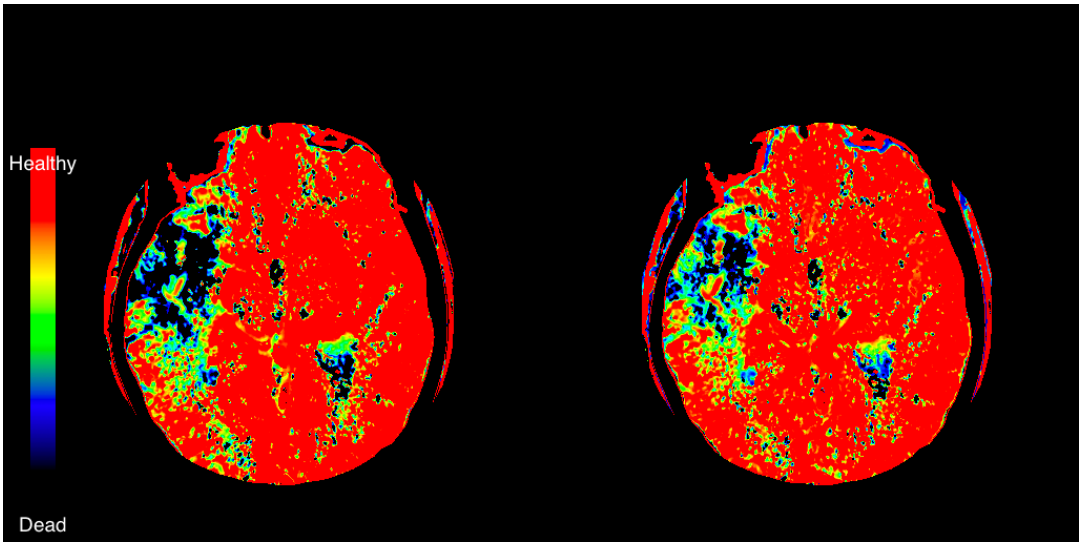
4.3.6 Improvement from Preprocessing

Figure 4.5 shows the important improvement the preprocessing mentioned in Section 4.2.2 brings. For both of the Pearson's and Spearman's correlation coefficients, without the preprocessing (Figures 4.5a and 4.5b), many fewer voxels can be marked up as healthy due to the impact of noise. For example, the right side of the brain should be healthy due to the information provided in hemodynamic maps (Figure 4.2), but many voxels in the right side of the brain can not be correctly identified and are therefore marked as green and blue. With the preprocessing, most of the brain voxels can be identified correctly as healthy tissue (red voxels). The preprocessing addresses the noise problem so well that it significantly improves the accuracy of the following correlation coefficient tests. The computational cost (running time) will be discussed for preprocessing in Section 4.3.14.

Figures 4.3 and 4.4 compare the raw signals with the referenced curve and for the same voxels, the GPR denoised signals with the reference curve, respectively. They are examples of how the preprocessing make differences in different tissue types more evident. The preprocessing enlarges both correlation coefficients in artery, grey matter and white matter.



(a) Pearson's without Preprocessing (b) Spearman's without Preprocessing



(c) Pearson's with Preprocessing (d) Spearman's with Preprocessing

Figure 4.5: The Importance of Preprocessing

The two subfigures 4.5a and 4.5b show segmentation results without preprocessing where subfigures 4.5c and 4.5d show the results of applying the correlations to denoised data using the GPR method. They correspond to Figures 4.6a and 4.6c which were prepared using preprocessing.

In this chapter, unless pointed out otherwise, all images are generated with denoising.

4.3.7 Correlation Analyzed Brain Maps

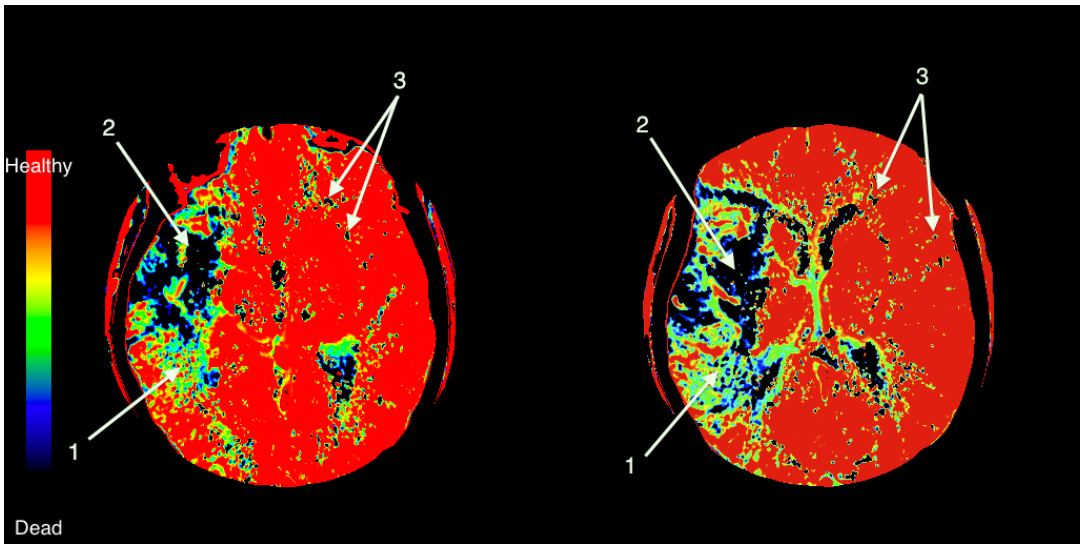
We then generated color maps showing the segmentation results. Using the correlation tests, the lesion area at the middle left side is very clear (as a dark area at the end of arrow 2) and can be easily spotted (Figure 4.6). As the CBF value is low in the white matter, it will be mapped to cold color in the colored CBF maps. However, a lesion area (ischemic stroke) can also lead to low CBF value. Although the CBF value in an abnormal area is lower than it in healthy white matter, their difference is small, which makes it hard to distinguish from each other visually (Figure 4.2)[4]. This situation also happens in other hemodynamic quantities, such as Tmax. The worst case is in the penumbra area in white matter, since the hemodynamic quantities in the penumbra tissue fall between dead tissue and healthy white matter, it is very difficult to distinguish the penumbra tissue in the hemodynamic quantity maps. Using correlation tests, they will be marked as potential lesion tissue (green and blue). The healthy white matter in Figure 4.6 is marked in red, which can be differentiated from abnormal tissue more easily.

If a voxel is marked as potential lesion tissue, it belongs to one of the three situations listed below:

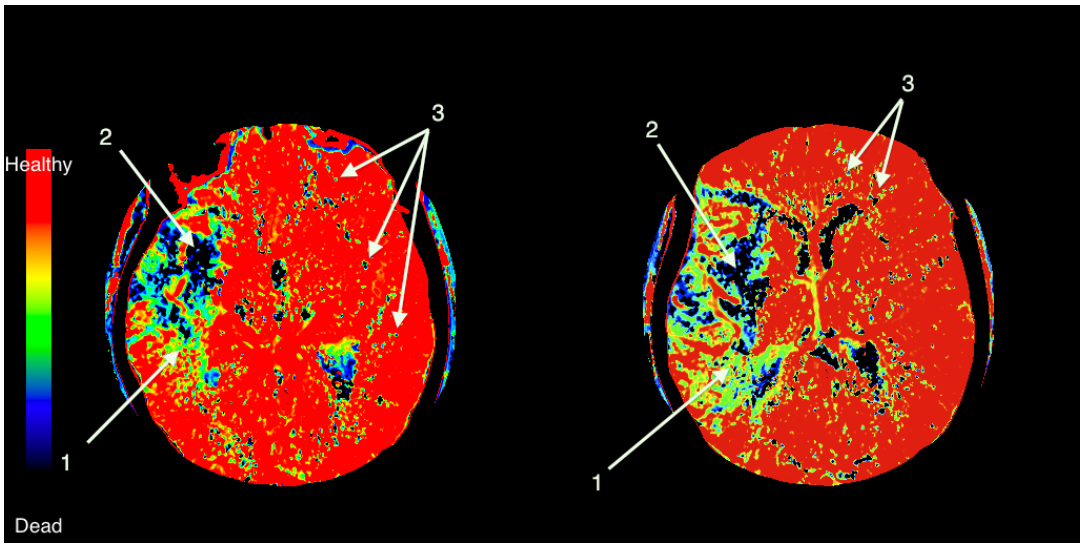
1. The penumbra tissues, which we sought to identify. They are affected by the stroke so that blood does not go through the tissue normally (each voxel either has lower blood volume or a larger delay), while they are not dead yet so there is still blood passing through them. This kind of abnormality results in a medium ρ/r value. Usually, if a large contiguous area is all marked with a low probability, it is possible that this area is penumbra that may still be saved. For example, at the bottom left side of Figure 4.6, the green and blue areas represent tissues at risk (the penumbra areas pointed by at arrow 1).

2. A false-positive error when the tissue is actually dead but its time-concentration curve coincidently has a shape that is similar to the healthy curve. These appear as lighter spots in the abnormal areas. For example, the spots in the dead area in

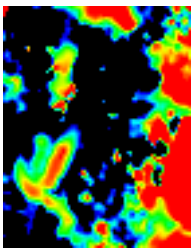
[4]The lesion areas in the CBF map appear as black, blue, green or even yellow areas.



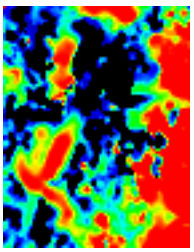
(a) Slice 1 - Pearson's Correlation Coefficient (b) Slice 2 - Pearson's Correlation Coefficient



(c) Slice 1 - Spearman's Correlation Coefficient (d) Slice 2 - Spearman's Correlation Coefficient



(e) Pearson



(f) Spearman

Figure 4.6: Results of Using Correlation Coefficient

Areas marked as red are healthy areas. If a black area is inside the brain, there is a high possibility that it is dead tissue. Arrows 1-3 point out the representation of the three possibilities of blue and green areas, which are penumbra tissue, false-positive dead tissue and false-negative healthy tissue, respectively. Figures 4.6e and 4.6f are the full-sized images for areas pointed by arrow 2 in Figures 4.6a and 4.6c, respectively.

the middle left side of the figures (pointed out by arrow 2) are possibly caused by false-positive errors.

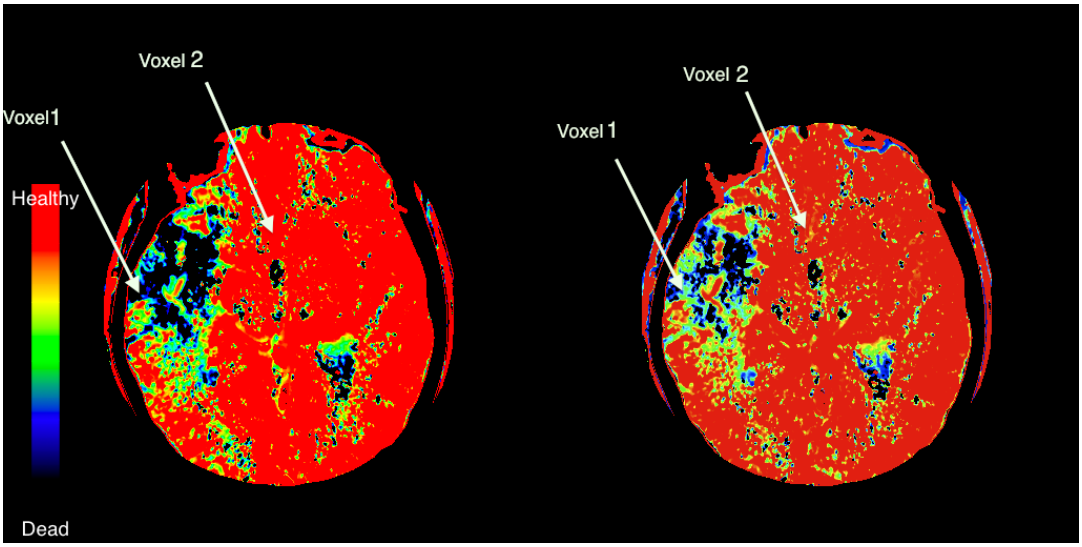
3. A false-negative error when tissues are actually healthy but the correlation tests can not detect this, because the tissues are impacted by noise and their low CNR. For example, the green and blue (or even black) areas at the right side of the brain are caused by false-negative errors (the spots pointed at by arrow 3). As it is possible that there are healthy tissues in infarct core areas and dead tissues in healthy areas, the second and the third situations may be either caused by false-positive and false-negative errors or be observations of as yet unexplained features of strokes.

Furthermore, our method not only points out which tissues have been affected, but also shows how serious the damage is. The damage from large to small will be designated by the colors from black (badly damaged) to blue, green, yellow and red (not damaged).

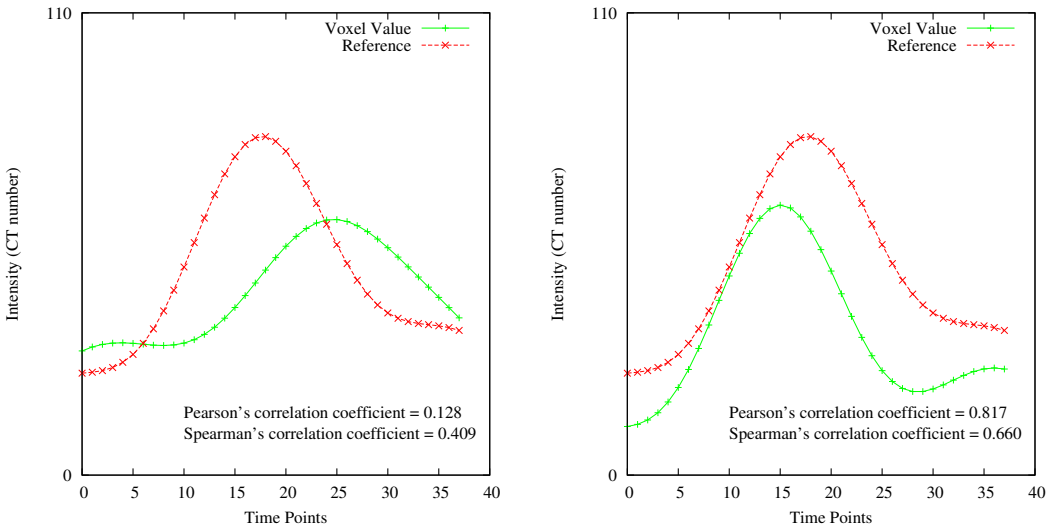
To conclude, correlation tests enhance the detection of the differences between healthy and abnormal tissues, especially in the white matter where the blood flow is low. This is important as it can reduce the possibility of failing to spot the lesion areas in some cases. Our correlation-based method improves the use of the time-domain data (temporal information) to enhance the discrimination between healthy and abnormal tissues. It makes fuller use of the time-domain data than previous methods. This makes available new and arguably better analysis results for diagnosticians and researchers. Correlation tests also deliver results with clearer delineation of abnormal areas.

4.3.8 Pearson's Correlation Coefficient versus Spearman's Correlation Coefficient

Figure 4.7 is an example of how Pearson's and Spearman's correlation coefficients can deliver different results. Voxel 1 is selected from a suspected lesion area with mismatched predictions in the different correlation methods; that is Pearson's correlation coefficient and Spearman's correlation coefficient deliver different predictions for this voxel. Voxel 2 is selected from the suspected healthy area with the same prediction from both of the correlation coefficients. Comparing Pearson's correlation coefficient (Figure 4.7a) with Spearman's correlation coefficient (Figure 4.7b), the former one states a larger area of dead tissue than the latter one; the Spearman's correlation coef-



(a) Pearson's Correlation Coefficient (b) Spearman's Correlation Coefficient



(c) Intensity Curve for Voxel 1 (d) Intensity Curve for Voxel 2

Figure 4.7: Pearson's Correlation Coefficient Vs. Spearman's Correlation Coefficient

Subfigures (c) and (d) are the tissue time-concentration curves for voxel 1 and 2. The Tmax value in voxel 1 is 26 seconds with an intensity value of 60.82 whereas the Tmax value in voxel 2 is 16 seconds with an intensity value of 141.80. They all come from denoised images.

ficient marks up many of these Pearson's-dead-tissues as tissues at risk. Figures 4.7c and 4.7d are a similar pair indicating how the two correlation coefficients lead to different results. Quantified results in each correlation coefficient will be stated in Section 4.3.12.

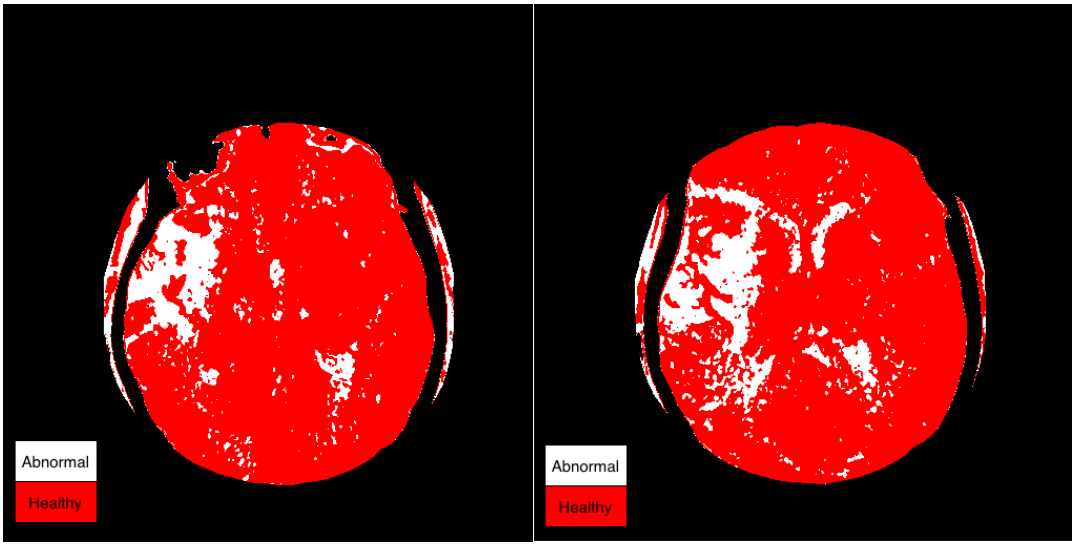
Figure 4.7c shows the tissue time-concentration curve from voxel 1 (indicated by an arrow). In this voxel, the Tmax is nearly 25 seconds, in comparison with the 16-second Tmax value in the reference voxel. There is abnormal tissues in this area. From the correlation coefficients, the Pearson's correlation r value is 0.128 and the Spearman's correlation ρ value is 0.409. From Equation 4.7 and 4.8, Pearson's correlation coefficient indicates that this voxel is a lesion at both the 0.01 and 0.05 significance levels, while the Spearman's correlation coefficient classifies the same voxel as healthy at both significance levels. Hence, for voxel 1, Pearson's correlation test classifies it as abnormal while Spearman's correlation test fails to detect the abnormality.

For the rest of the brain, there are more yellow and blue areas but less black areas in the Spearman's correlation coefficient (Figure 4.7b) than Pearson's (Figure 4.7a). Voxel 2 (Figure 4.7d) from the artery is an example. The Tmax for this voxel is the same as for the reference voxel, which is 16 seconds. For Pearson's and Spearman's correlation coefficient ρ or r values for this voxel are 0.747 and 0.869, respectively.

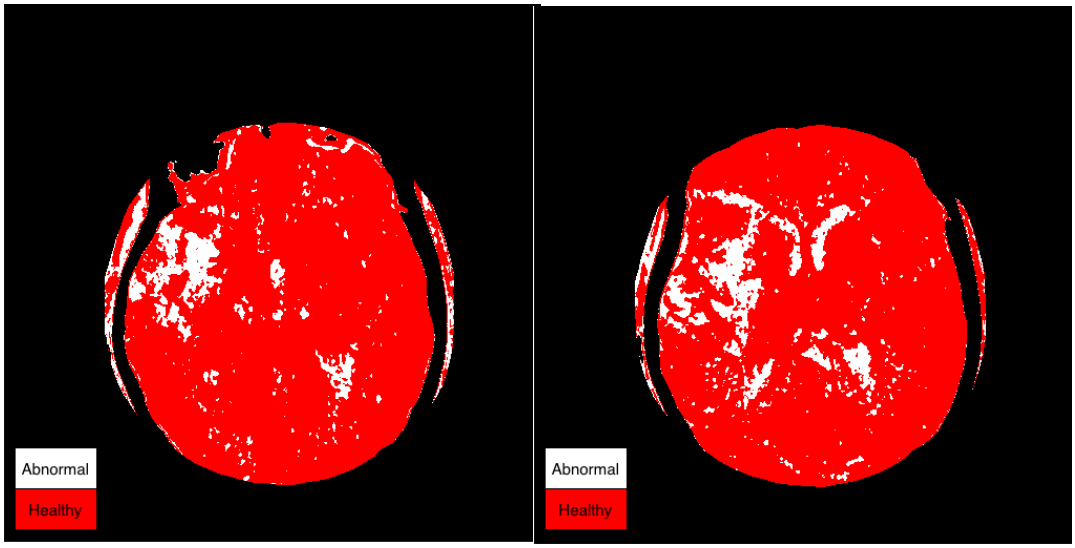
4.3.9 Student's T-Test Bit Maps

Figure 4.8 shows bit-maps generated using Student's t-test to threshold results from correlation coefficient tests. It uses statistical analysis to help us distinguish abnormal tissues from the healthy tissues. It can reduce the false-negative errors in the area near the brainstem. It can also help us point out the region of the dead tissue.

However, as the Student's t-test provides only two possible outputs (healthy tissue or abnormal tissue), it is not possible to identify penumbra tissues. Compared to the methods that directly map correlation values to color, the advantage of this method is that it uses a statistical method to help determine whether or not measured tissues are abnormal; the disadvantage of t-test method is that it cannot indicate the fitness level well, which reveals how certain the correlation test is that the target is healthy (or abnormal). Another drawback of using t-test is that it can not distinguish the dead and penumbra tissue.



(a) Pearson's + Student's T-Test



(b) Spearman's + Student's T-Test

Figure 4.8: Results of Student's T-Test

These figures are results of Student's t-test. They are also from the same patient as the Figure 4.2. The red areas indicate healthy areas and white areas are lesion areas. The given significant level is 0.05.

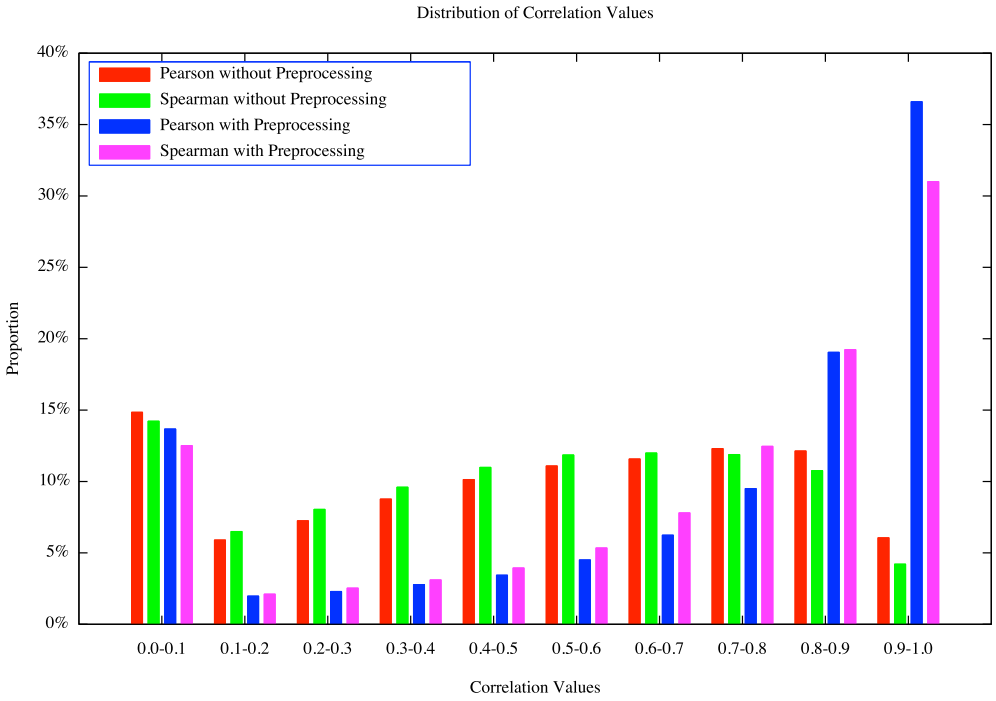


Figure 4.9: Distribution of Correlation Values

This histogram illustrates the distribution of correlation values using both Pearson's and Spearman's correlations with or without preprocessing. The proportion is obtained by dividing the number of voxels, which have correlation values in the given value range, by the overall number of valid (non-background) voxels.

4.3.10 Results from All of the Subjects

4.3.11 Results from All of the Subjects

In Table 4.1, it can be found that the correlation tests work well. For subjects with lesion — subject 1 (lesion areas at left side), subject 3 (lesion areas at left middle side), subject 4 (lesion areas at top right side), subject 6 (lesion areas at bottom) and subject 7 (lesion areas at left) — both Pearson's correlation test and Spearman's correlation test are able to spot the abnormal tissues and to determine their boundaries. The correlation tests can also provide the suggestion that no notable lesion areas are present in subjects 2, 5, 8 and 9.

Ten is a small number of subjects in particular with only two slices of data for each patient. So there may be misleading sampling error and a bigger study is necessary to confirm the results.

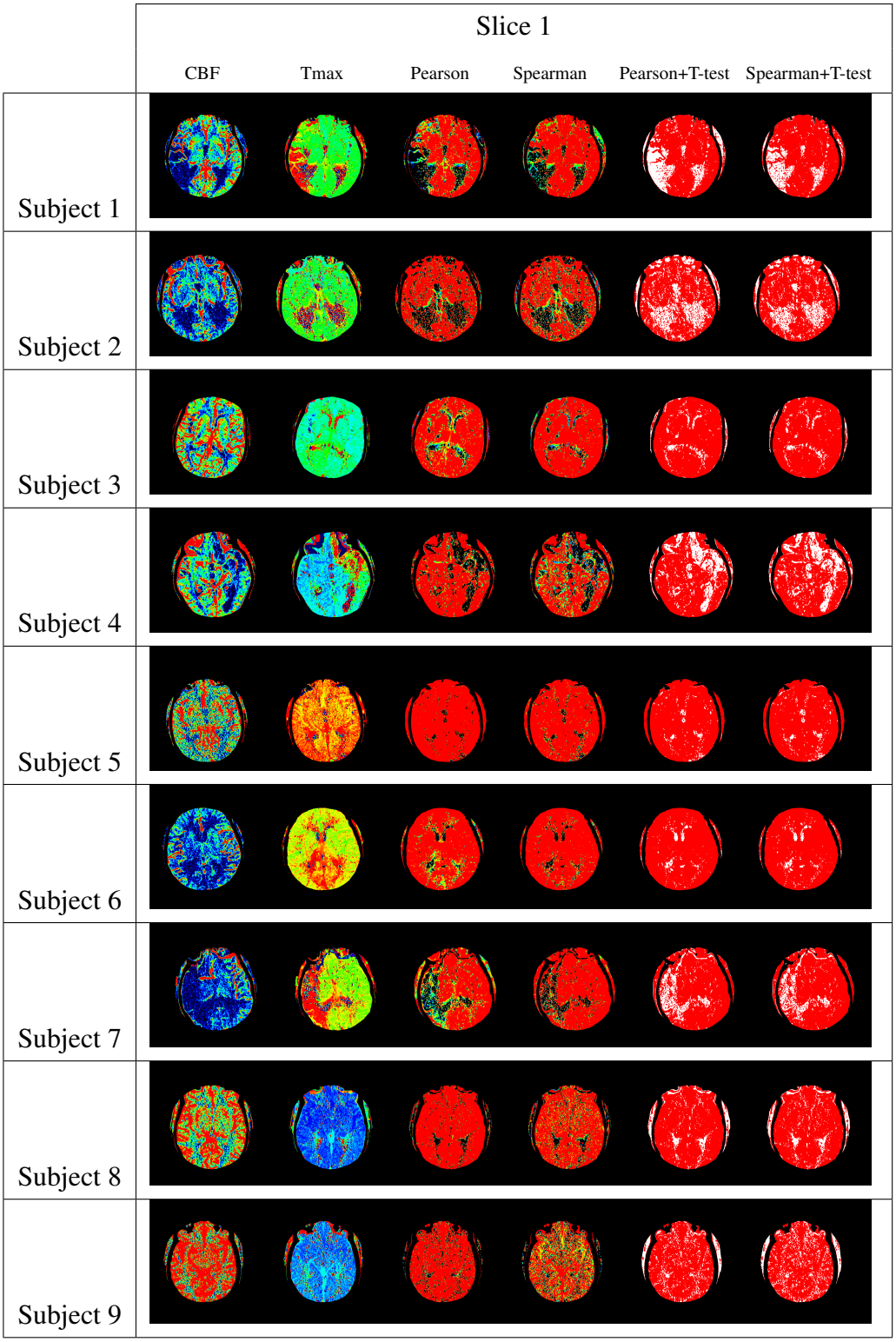


Table 4.1: Results from All of the Subjects (Slice 1)

This table illustrates the results of correlation methods for the remainder of the nine subjects. For each slice, a CBF map and a Tmax map are included as references; the results of both of the Pearson's correlation test, Spearman's correlation test and Student t-test are presented. For each subject, both of the slices are displayed. Images for slice 1 are in this table and images for slice 2 are in Table 4.1 continued. They are using the same color map-key as Figures 4.2, 4.5 and 4.7.

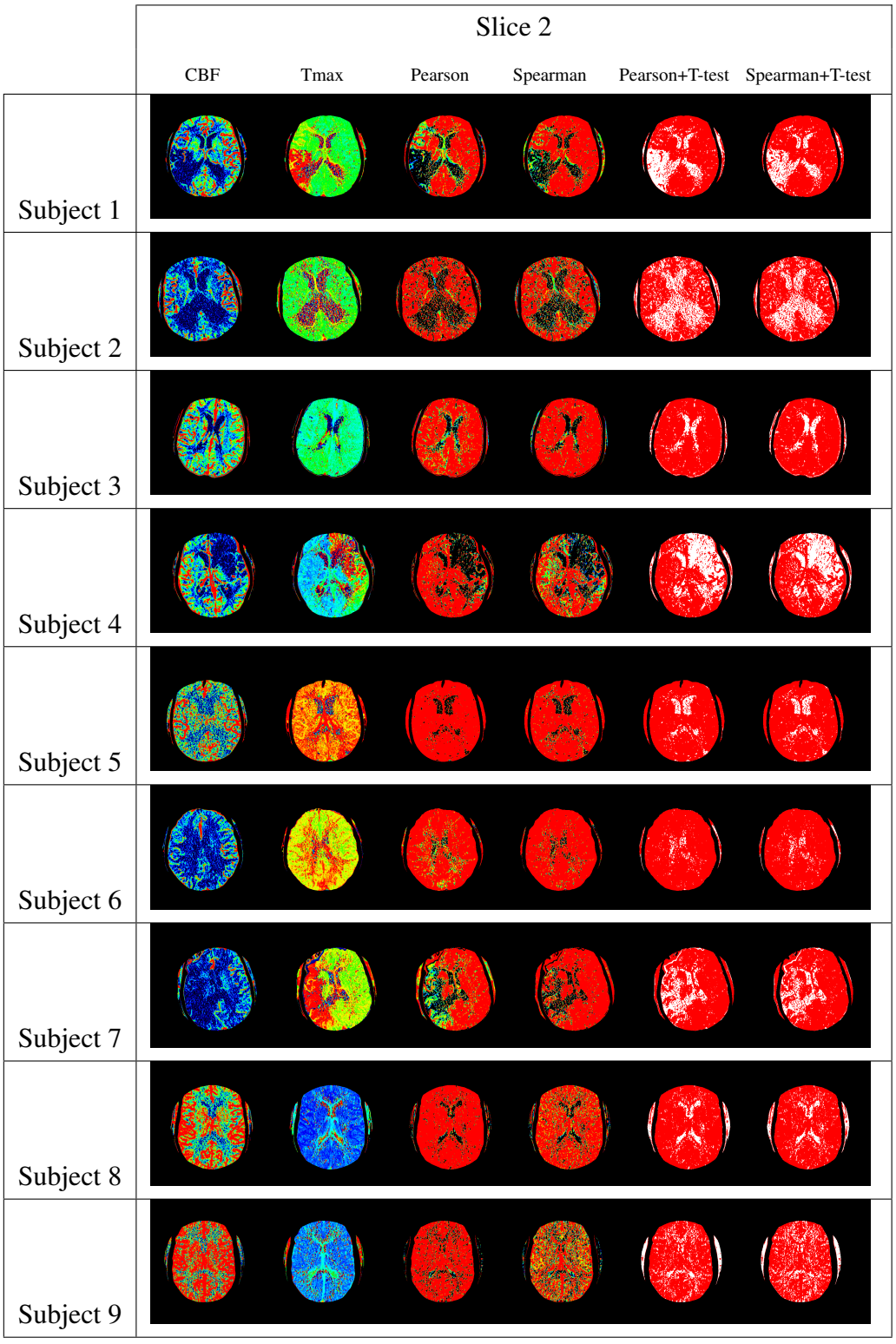


Table 4.1 continued: Results from All of the Subjects (Slice 2)

This table contains images for slice 2 for all the nine subjects.

4.3.12 Distribution of Correlation Values

Figure 4.9 indicates the differences resulting from preprocessing in the distribution of the correlation values for both Pearson's and Spearman's correlation coefficients used in our experiments. Without preprocessing, both correlation tests result in a higher proportion of values in low- and medium-value ranges [0.0-0.7], compared with the proportion values with preprocessing. Furthermore, only 6% and 4% of voxels have correlation values larger than 0.9, for Pearson's correlation coefficient and Spearman's correlation coefficient, respectively. The values are increased to 37% and 31% when preprocessing is used.

Figure 4.9 also shows that Pearson's correlation coefficient (with preprocessing) accepts more voxels than Spearman's correlation coefficient (with preprocessing) in the values ranging from 0.9-1.0. For both correlation coefficients, 75.8% (with the difference $\leq$0.1%) of voxels are in the medium- and high-value ranges [0.5-1.0].

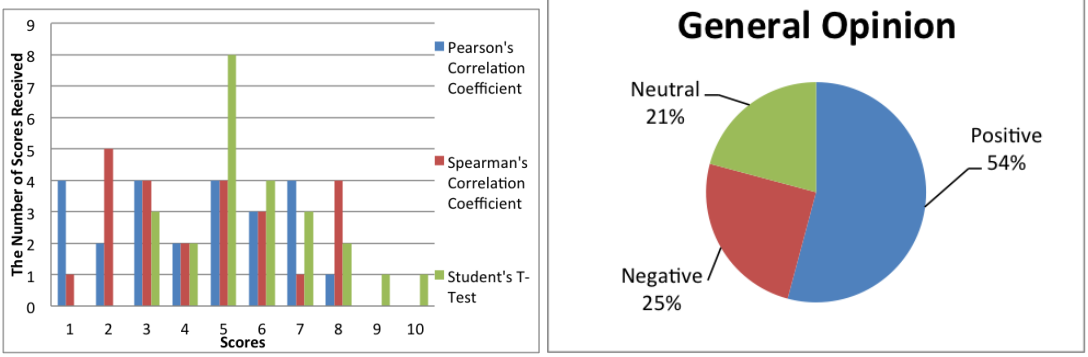
4.3.13 Experts' Opinions

Experts' opinions are also considered in our experiments. It is the same study as mentioned in Section 3.3.9. The questionnaire can be found in Appendix A (Questions 14 to 15).

Groups of images were shown to 12 experts (including neurologists, radiologists, stroke physicians and geriatricians) with an average of 9.5 years of experience. Within each test group, there was one CBF map, one Tmax map and three segmentation maps using Pearson's correlation coefficient, Spearman's rank correlation coefficient and Student's t-test[5]. For example, Figure 4.2a, Figure 4.2c, Figure 4.6a, Figure 4.6a and Figure 4.8a formed a group.

Experts were asked to give their view on the usefulness level (from 1 to 10, where 1 means ideal and 10 means no value) of the segmentation using CBF map and Tmax map as references. As illustrated in Figure 4.10, the experts considered Pearson's correlation coefficient and Spearman's rank correlation coefficient to be almost equivalently accurate with average (standard deviation) scores of 4.25 (2.18) and 4.5 (2.20), respectively. While Student's t-test received worse scores which were 5.7 on average, with a standard deviation of 1.79. More than half (54%) of respondents thought

[5]As using Student's t-test to threshold the Pearson's r value and Spearman's ρ value leads to similar results, only Pearson's correlation coefficient with Student's t-test was used to keep it simple. The Student t-test mentioned in this section indicates the method that use Student's t-test to threshold the Pearson's r value.



(a) Detailed (b) General

Figure 4.10: Questionnaire Results

Figure 4.10a shows the distribution of the feedback scores for the three given method. It indicates the number of scores received for different methods. Figure 4.10b illustrates people's attitude toward different segmentation results.

our segmentations are accurate and are helpful for clinical decision-making, while one quarter of respondents hold negative opinions and about one fifth of them hold neutral opinions.

4.3.14 Running Time

Our experiments were performed on an *Intel*® *Xeon*® CPU, which is the same CPU used in Section 2.3.1.1. It contains two cores but only one of them is used. The frequency of each CPU core is 3 GHz. The overall CPUs memory is 8 GB and their cache size is 4 MB each. The CPU's single precision floating point performance (peak) is 2.4 GFLOPS each core.

The running time is measured by the arithmetic mean of ten repeated tests. The dataset size is $512 \times 512 \times 2$ with 36 time intervals[6], which is the same as mentioned in Section 2.3.1.4. As stated in Section 3.3.10, the preprocessing using Gaussian process regression to reduce noise takes about ten seconds. Correlation tests using Pearson's correlation coefficient only take one second to run. Spearman's rank correlation coefficient takes 11 seconds to execute, because it takes an extra ten seconds to sort the intensities of the time-concentration curve for each voxel compared with Pearson's correlation coefficient.

In our implementation, bubble sort ($O(n^2)$) is used when calculating Spearman's rank correlation coefficient. However, since the range of the intensity values is known,

[6]It is 40 time intervals in the input, but four of them are removed during preprocessing.

it is possible to use bucket sort ($O(n+k)$) to improve the performance[7]. Besides, since the correlation tests for different voxels are independent, it is possible to use data parallelism to speed up the process. However, considering that the overall running time is less than half a minute for both correlation coefficient methods, we did not exploit parallelism in the measurements. But there is great potential to improve the performance further.

4.4 Summary

We conducted an investigation to test whether correlation in the time-domain would yield useful additional information about the tissue in each voxel of a CT brain perfusion imaging scan. Three statistical tests were investigated: Pearson's and Spearman's correlation coefficients and these in combination with a Student's t-test. These were applied to both raw and denoised time-domain data obtained from the ten subjects introduced in the preceding chapter.

The measurements and visual inspections of the results of the experiments suggested that our method makes fuller use of the time-domain data than previous methods. This makes available new and arguably better analysis results for diagnosticians and researchers.

We also measured the computational costs of this tissue classification on the equipment we used, a standard $Intel^{®}$ $Xeon^{®}$ CPU. It costs 1 seconds and 11 seconds for Pearson's and Spearman's correlation coefficients, respectively[8].

We then conducted a study to evaluate results with expert opinion via a questionnaire. More than half of respondents thought our methods are accurate and helpful for clinical decision-making, while 25% of respondents hold negative opinions and 21% of them hold neutral opinions.

We conclude that based on these initial experiments, perfusion source images can be analyzed with good accuracy in a reasonable time without the necessity of expert intervention during image processing. Our approach enlarges the usage of perfusion source images and produces comparable results to hemodynamic parametric maps. At the level of individual voxels, correlation tests successfully enhance and indicate the difference between healthy and abnormal tissues, especially in the white matter where the blood flow is low. This is important as it can reduce the possibility of failing to

[7] where n is the number of time intervals and k is the number of buckets in bucket sort.

[8] The preprocessing takes ten more extra seconds for both of the two correlation coefficients.

spot the lesion areas in some cases. Therefore healthy, dead and penumbra tissues can be distinguished more clearly and more easily. Over the entire image, the boundary of lesion areas can be drawn more sharply compared with CBF and Tmax maps. If only dead tissue is required to be identified, using Student's t-test to threshold correlation values from correlation tests is a good solution. This tissue classification method we have pioneered will have significant potential to improve the care of patients with acute ischemic stroke.

Our interpretation of the results could be in error due to the small number of datasets used. This clearly warrants further investigation on a broader spectrum of subjects and with a larger cohort of potential users.

Chapter 5

Summary and Conclusions

In this chapter, I will summarize the work presented in my PhD research, state its contributions and limitations, and conclude with ideas for further research.

5.1 Conclusion

Brain perfusion weighted images acquired using dynamic contrast studies have an important clinical role in acute stroke diagnosis and treatment decisions. The purpose of my PhD study is to develop novel methodologies for improving the efficiency and quality of brain perfusion-imaging analysis so that clinical decisions can be made more accurately and in a shorter time.

The first issue addressed was the acceleration of the algorithms used to generate hemodynamic maps. It proved possible, using GPGPUs, to achieve substantial speedups while retaining the quality of generated hemodynamic maps. The next two investigations sought to make better use of the information latent in the time-domain of CT perfusion scans. The first of these showed that Gaussian process regression (GPR) could significantly enhance the quality of derived data at modest computational cost. The second of these then used that denoised data to improve the discrimination between healthy, penumbra and dead tissue. Each of the stages of this investigation are now revisited below, with an assessment of their impact and reliability.

5.1.1 Accelerated Generation of Hemodynamic Maps

A stroke is a medical emergency in which time is critical for a stroke patient; the sooner the treatment occurs, the less damage that will be caused to a patient's brain (⇝1.11).

Table 5.1: Performance Improvement

Data Size ($Dim1 \times Dim2 \times Dim3 \times time$)	GPGPU Speedup	OpenMP Speedup	MPI Speedup
$128 \times 128 \times 22 \times 80$ (MRI Image Size)	3.75	2.21	3.42
$128 \times 128 \times 11 \times 44$ (CT Image Size)	5.56	2.26	3.84

When the local AIF technique is used (⇝1.9), which requires the decomposition of thousands of AIF matrices, the performance is poor in terms of running time. In order to reduce analysis processing time and increase the possibility of using local AIF in clinical diagnosis, we need to accelerate the analysis process.

GPGPU (⇝2.1.4), a shared-memory parallel based architecture, is applied in our methods. The key idea is to separate the computational tasks into voxel-based small tasks, distribute them to different GPU threads and obtain performance improvements from data parallelism. Other performance optimization techniques, such as data reconstruction, are also used in our methods. Our method does not change the quality of the hemodynamic maps. In other words, our GPGPU implementation delivers the same results as the original serial implementation's results.

We have analyzed computational complexity and feasibility for every individual step in the perfusion imaging processing and applied different parallelism methods based on the analysis (⇝2.2.3). *Data reorganization* and *Denoising & reorganization* steps use lower-level parallelism in which each GPU thread is in charge of the task for one voxel and one time point. In the *Deconvolution* step, upper-level parallelism is used, by distributing deconvolution (contains all of the time points) for different voxels to different GPU threads. Theoretically, for both these steps, parallelization enables all of the voxels in a scan to execute concurrently. However, due to the limitation of local memory, parallelization for the *Deconvolution* step can only be performed one slice at a time (⇝2.2.6). The *Deconvolution* step is the bottleneck for perfusion-imaging analysis, although the speedup factor is more than one hundred for both the *Data reorganization* and *Denoising & reorganization* steps, the overall performance speedup factor is limited by this bottleneck (⇝2.3.2).

Table 5.1 shows the speed up achieved by different parallelization implementations, compared with the original serial implementation (⇝2.3.3). The overall processing time is reduced from six minutes to 65 seconds for our CT test dataset and from 35 minutes to less than ten minutes for our MRI test dataset. For the CT and MR data sizes given in the experiment, our implementation has gained a speedup factor of 5.56 and 3.75, respectively. The speedup also depends on the CUDA configuration parameters

which determine how tasks are assigned to GPU cores. Our experiments also show that the four core CPU parallel implementation using OpenMP and MPI gains a speedup of 2.21 to 3.84 depending on the data size, which is smaller than the improvement brought about by GPGPU implementations.

Since the improvement of technology will increase the resolution of perfusion imaging and the number of available sampling time points, the computational cost for perfusion imaging analysis will keep rising. Ordinary GPUs have evolved into highly parallel, multithreaded, many-core processors with tremendous computational power. As a result, GPUs become more and more suitable for large-scale computational tasks and have great potential to be used to solve complex perfusion imaging analyses in the future.

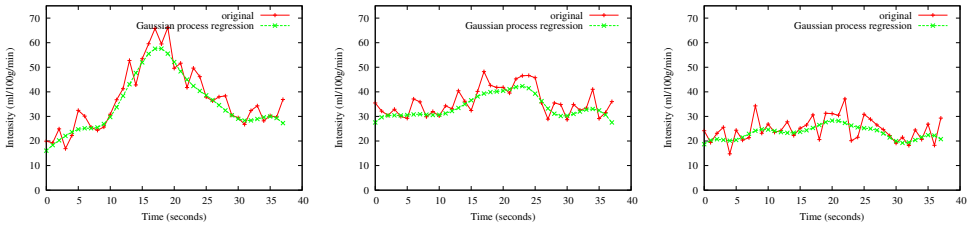
The major limitation in this part of research is that: the selection of platforms to perform GPU and CPU parallel implementations are tricky (⇝2.3.1.1). The GPU and CPUs used in the experiments are both ordinary ones with similar price. But it is still possible that our experiments are not a fully fair trial. For example, we may have missed an optimisation opportunity in one of our implementations ultimately; it will also be important to consider the impact on maintainability of using GPGPU-based methods.

We conclude that parallelization using GPGPU can dramatically reduce the running time of perfusion imaging analysis based on local AIFs. The GPGPU implementation is superior to serial implementation and to both CPU parallel implementations. It also increases the possibility of using local AIFs in clinical diagnosis.

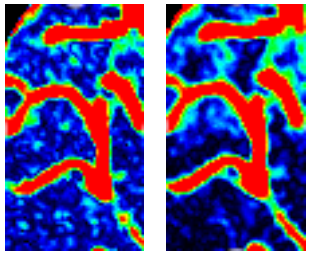
5.1.2 Reducing the Effects of Noise

Since X-ray radiation increases the risk of inducing cancer, CT scanning is constrained by a tradeoff between image quality and the amount of radiation exposure to the patient. Traditional noise reduction methods are usually based on three dimensional information, which does not fully utilize the time-domain of perfusion images.

Our approach uses a noise reduction method based on Gaussian process regression (GPR), which benefits from the 4D (temporal) information in tissue time-concentration curves (⇝3.2.1). It uses the continuity property in the tissue time-concentration curves and performs as a temporal denoising filter. The smoothness level of denoising output is determined by hyperparameters in the covariance function and the hyperparameters are adjusted based on derivatives of the features of perfusion images. We also



(a) Artery (b) Grey Matter (c) White Matter



(d) Raw (e) GPR

Figure 5.1: Results of GPR

developed a noise reduction method, called multiple observation Gaussian process regression (MGPR) (⇝3.2.3). It works as a combination of a spatial decimation filter and a temporal filter. MGPR reduces the resolution of the result but can benefit being from both a spatial decimation filter and a temporal filter.

The quality and reliability of our methods are examined in four ways: by considering whole-brain summary statistics, by examining individual voxel's tissue time-concentration curves and hemodynamic maps, and by a usability study.

1. The results of GPR without spatial decimation show a 99% improvement in CNR over raw data. Our spatial decimation based GPR also shows considerable improvement compared with the spatially weighed mean filter. For a 3×3 spatial decimation, our MGPR and MEAN & GPR achieve 102% and 78% higher CNR than raw data respectively. This should be compared with the 18% improvement achievable by simply using a weighted mean filter and the 11% improvement obtained by using a TIPS bilateral filter. (⇝3.3.7)

2. Considering the individual voxel, GPR denoised tissue time-concentration curves have much smaller oscillations than raw data (Figures 5.1a to 5.1c), which helps to distinguish parameters, such as the baseline value and Tmax, more easily and with better accuracy. (⇝3.3.5)

3. For hemodynamic parametric maps (Figures 5.1d and 5.1e), our GPR methods

provide a much better solution with clearer edges and more detailed information achieved by the other approaches we studied. These results show that Gaussian process regression based methods handle noise better than comparable techniques used. (⇝3.3.8)

4. In the usability study (questionnaire), results without denoising (raw images) received a very low ranking score of 4.65 ($\pm$0.56)[1]. GPR and MGPR are much preferred with scores of 3.29 ($\pm$0.60) and 3.00 ($\pm$0.64), respectively. (⇝3.3.9)

The running times for GPR and 3×3 MGPR are about 12 seconds and 100 seconds, respectively (⇝3.3.10). Our methods, improving the quality of output hemodynamic maps in reasonable time, make them have potential application for clinical diagnosis and brain research.

The results are evaluated based on a small number (ten) of subjects and only two slices per subjects were available, so there may be misleading sampling errors.

We believe our methods are the first to fully utilize the temporal information to reduce the noise in perfusion source images. Our methods dramatically improve signal-to-noise ratio and then enhance the quality of hemodynamic maps. They make diagnosis easier by providing better boundary and detail information.

5.1.3 Discriminating Healthy, Penumbra and Dead Tissue

Since brain perfusion source images contain more information than hemodynamic maps, a good utilization of the source images can lead to better understanding of scan results. Our method makes use of the perfusion source images and identifies lesion areas automatically using a correlation test.

We have developed a classification method using information contained in the time-domain of perfusion source images. We have investigated correlation-coefficient-based methods to distinguish abnormal tissues from healthy tissues (⇝4.2). Three statistical tests were investigated: Pearson's and Spearman's correlation coefficients and these in combination with a Student's t-test.

At the level of individual voxels (⇝4.3.5), correlation tests successfully enhance and indicate the difference between healthy and abnormal tissues. For example, as CBF values in white matter and abnormal tissue are both low, the boundaries separating white matter and abnormal tissue are hard to determine. As a result, our methods

[1] Scoring from 1 to 6, where 1 is the best and 6 is the worst.

can reduce the possibility of failing to spot the lesion areas in some cases. Over the entire image (⇝4.3.7), the boundary of lesion areas can be drawn more sharply compared with CBF and Tmax maps. If only dead tissue is required to be identified, using Student's t-test to threshold correlation values from correlation tests is a good solution (⇝4.3.9).

We conducted a study to evaluate results with expert opinion via a questionnaire (⇝4.3.13). More than half of the respondents think our methods are accurate and helpful for clinical decision-making, while 25% of respondents hold negative opinions and 21% of them hold neutral opinions.

Our interpretation of the results could be in error due to the small number (ten) of datasets used. This clearly warrants further investigation on a broader spectrum of subjects and with a larger cohort of potential users.

We conclude that based on our initial experiments, perfusion source images can be analyzed with good accuracy in a reasonable time (several seconds) without the necessity of expert intervention during image processing. Our approach enlarges the usage of perfusion source images and produces comparable results to hemodynamic parametric maps. No additional information is required as input to our methods, in other words, our approach uses the same input as hemodynamic based methods. But our approach automatically delivers a suggestion of classification to differentiate healthy tissues from penumbra and dead tissues. Such automated classification opens up a number of methods for generating summative quantitive reasons, such as the relative volumes of tissue in each category.

5.1.4 Integrated View

Every part of my PhD research strives toward the same goal - to deliver better perfusion information in shorter time. Thus, the three parts of my PhD research are inseparable. Parallelization using GPGPU can not only speed up the perfusion imaging analysis, but can be also used to reduce the running time of our noise reduction methods and our automatic classification methods. Both our noise reduction methods and classification methods are based on the temporal information in the perfusion source images. The denoised result also improves the classification result.

5.2 Further Work

While my PhD research has tried to solve perfusion imaging related performance and accuracy issues, there remain many open questions for further research:

- Our noise reduction method using multiple observation Gaussian process regression can reduce the noise using a combination of 2D image information and tissue time-concentration curves. As the resolution in the third dimension is 1/12 of that within a slice, i.e. for the scanner used in the experiments, 1mm $\times$ 1mm within slice and 12mm in the direction of the stack of slices, the 2D information (slice based) can be improved to 3D (volume based) which may lead to extra improvement. However, new scanners might provide better resolution, making 3D plus temporal filters more tractable.

- Conducting the evaluation of the methods in Chapters 3 and 4 on a larger sample of patients to confirm that the gains are sustained, e.g. when data comes from a variety of scanners. If possible, this would also explore examples with more slices available.

- Conducting more extensive usability trails. In particular, investigating with a larger and more diverse cohort of potential users, the utility of (a) the improved derived data, and (b) the choice of presentation methods. Ideally, this cohort should be sufficiently extensive that different responses from different clinical roles can be detected and the value of the information in clinical training can be assessed. Such more extensive trials are a necessary precursor to wider adoption in clinical practice.

- It is common to have clusters of processors available, with or without GPUs per node. This suggests the possibility of further acceleration with GPU and sufficient acceleration with CPU by investigating the speedup with multiple nodes (multiple CPUs) and multiple (CPU + GPU) nodes.

- In our noise reduction method mentioned in Chapter 3, only the estimated value is used. However, Gaussian process regression also provides a confidence interval for every estimated value. It is possible to investigate whether presenting confidence information in conjunction with hemodynamic maps is helpful to researchers and diagnosticians. There would be an adjunct research issue of how best to present it.

- Our method, MGPR, is very promising in its design, but it did not work very well in our experiments. The main reason is that it can not handle the boundaries of different tissue types well, so it blurs the image in some specific areas. A solution to this is to use a correlation test to determine the boundaries (inferring that two low-correlated voxels probably belong to two different tissue types) and exclude voxels from different tissue types from the spatial decimation.

- The TIPS filter noise reduction method uses the sum of least squares to measure the similarity between two different voxels, but a correlation coefficient can be a better evaluation of the similarity as explored in Chapter 4. The TIPS filter may be improved if it makes use of such a correlation coefficient.

Building on much previous research into perfusion imaging we have taken significant steps towards making better use of all of the data, particularly the time domain. These steps open many new opportunities for observing and analysing perfusion phenomena. These should be explored further in the brain but they may also be relevant for other organs and morbidities.

5.3 The PhD in Perspective

This PhD research is an interdisciplinary research. As a computer scientist, it is very important to work with colleagues in another discipline, such as opportunities from embedding in Informatics and in BRIC (Brain Research Imaging Centre Edinburgh). The training I had in EPCC (Edinburgh Parallel Computing Centre) regarding parallel computing and in China regarding mathematical skills were also very important to understand and find answers for the PhD's goals.

The PhD study is worthwhile and I enjoyed it very much.

Appendix A

Questionnaire and Results

The questionnaire is used to evaluate the results from both of the chapters *Noise Reduction Using Gaussian Process Regression* and *Automatic Lesion Area Detection*.

The original results are filled (in red) in the form in order to make the result intuitive. The order of images in each group is disrupted to prevent inertia of thinking. The names of the method that images represent are not available to the participants, they are added in the questionnaire in order to improve the readability of the results after the survey.

Anything in RED in the questionnaire is not accessible to the participants.

Study of Brain Perfusion Image Processing

Thank you very much for taking the time to do this questionnaire.
This questionnaire is part of a study of my work on perfusion imaging. This work is about trying to improve methods for displaying the information from perfusion imaging. One of the problems in perfusion images is background noise and I have been working on methods to reduce the noise using Gaussian process regression. The other part of this work is to automatically detect the lesion areas.
This is part of my PhD research, supported by SINAPSE, the Brain Research Imaging Centre and Prof. Joanna Wardlaw.

All of the images are CT perfusion imaging results from three stroke patients (one and a half hours to five and a half after stroke).
Each page shows images from one slice of one single patient that has had different denoising methods.

Please help me to get to know a little more about you:

1. You are a: ☐ medical physicist, ☐ radiographer, ☐ neurologist, ☐ geriatrician, ☐ stroke physician, other: _____

2. You have ____ years experience in this field.

3. You are doing this questionnaire on: ☐ Screen or ☐ Paper

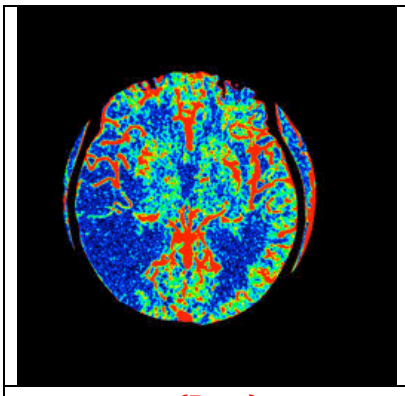
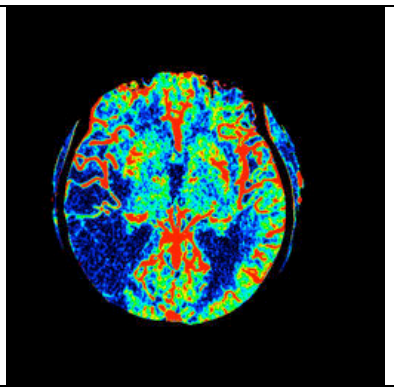
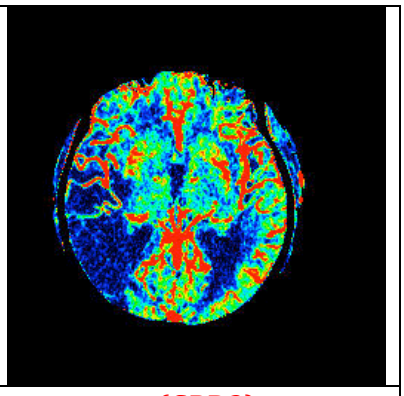
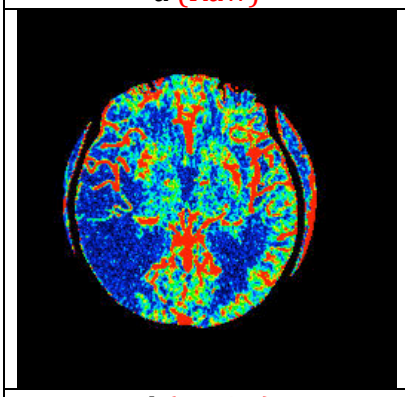
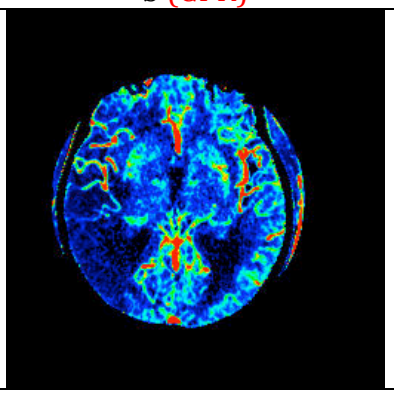
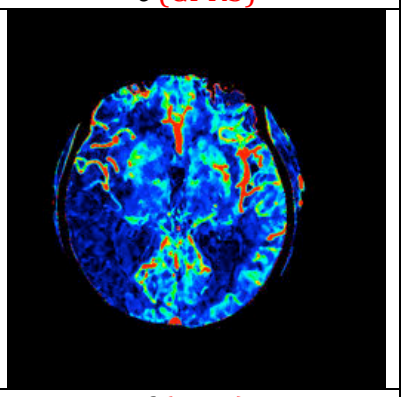
4. I am not going to identify you in the results, but if you want to see the results, please leave your e-mail address here: _____

Please, if at all possible, complete this by Jan 6th 2012, as need the results for my research. We will try to use late results.

If you would like me to send you a printed color copy of this questionnaire, please email me at F.zhu@ed.ac.uk, giving your postal address.

If you completed this questionnaire on your computer, please email it to:
F.zhu@ed.ac.uk
If you completed it on paper, please email it to:
Fan Zhu,
Room 5.24, Informatics Forum,
10 Crichton Street
Edinburgh, Midlothian EH8 9AB
UK

1. CBF

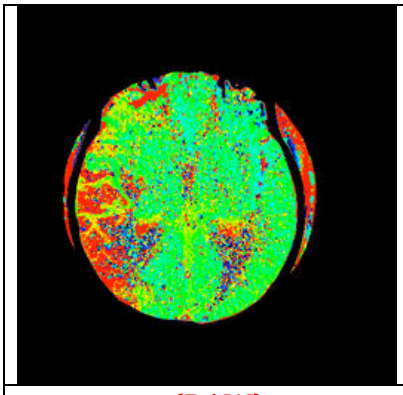
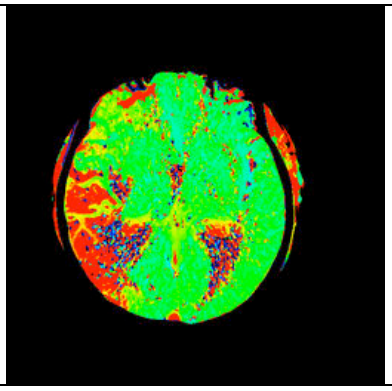
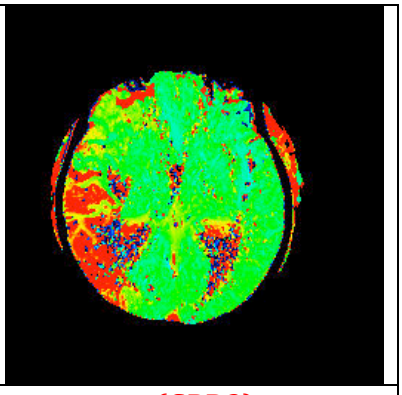
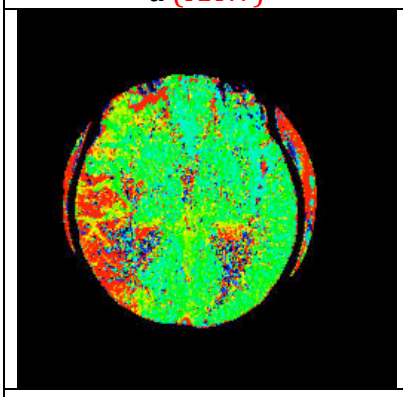
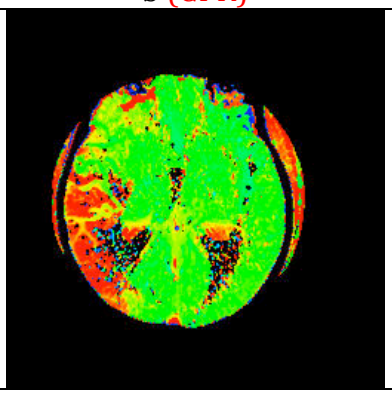
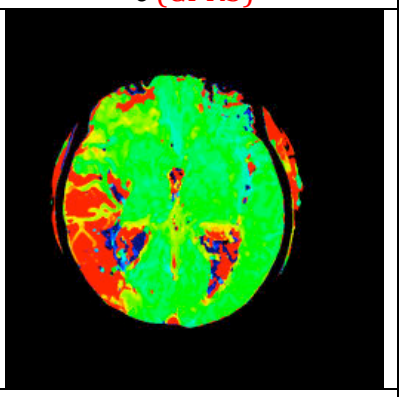
		
a (Raw)	b (GPR)	c (GPR3)
		
d (MEAN)	e (MF)	f (TIPS)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is.

a	4.42
b	2.27
c	2.08
d	3.33
e	4.50
f	4.00

Comments or reasons if they apply:

2. TTP

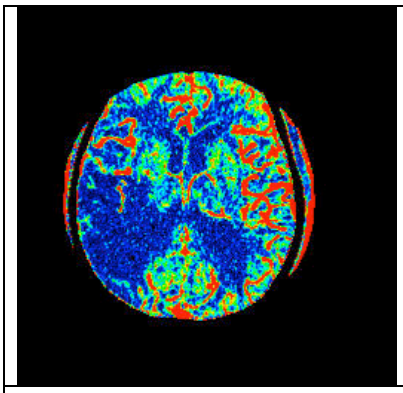
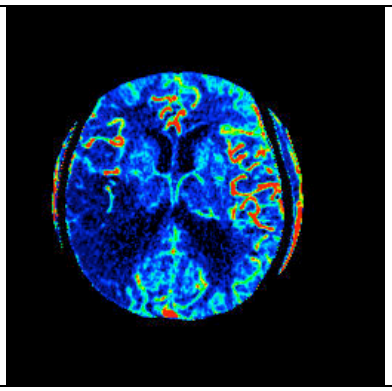
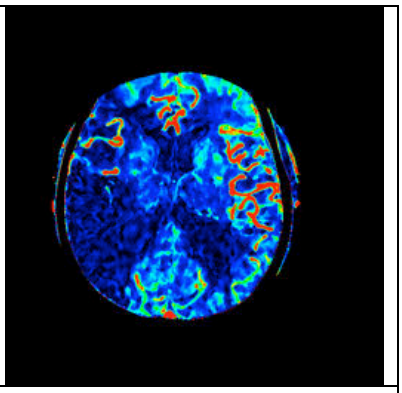
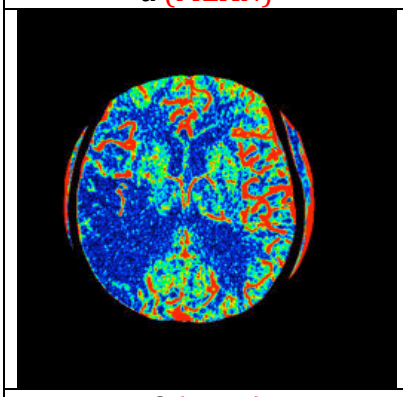
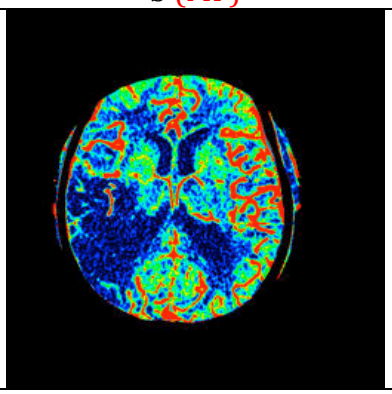
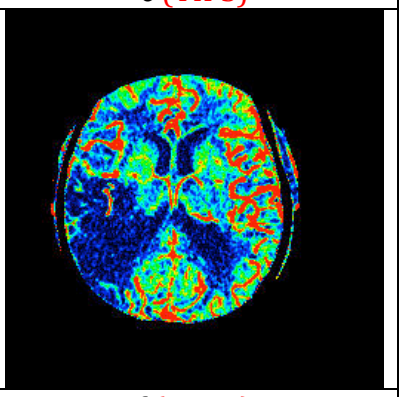
		
a (RAW)	b (GPR)	c (GPR3)
		
d (MEAN)	e (MF)	f (TIPS)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	5.25
b	3.75
c	3
d	5.58
e	2.33
f	1.08

Comments or reasons if they apply:

3. CBF

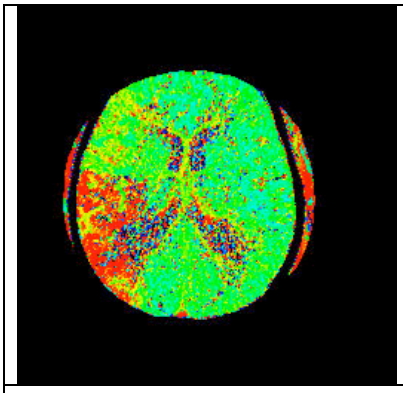
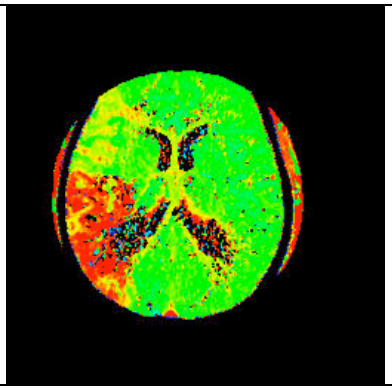
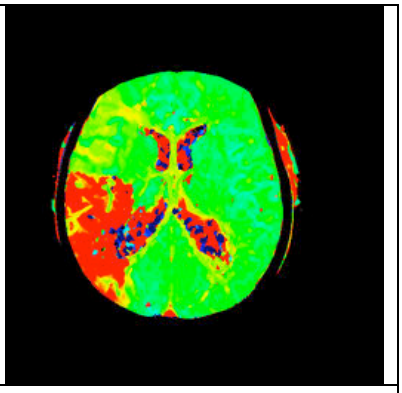
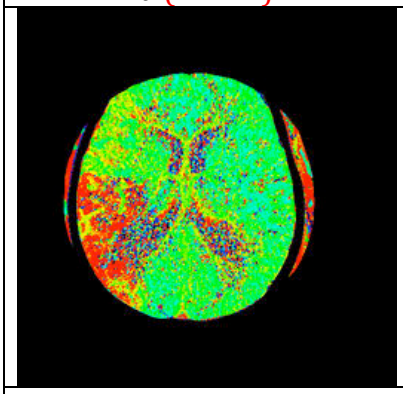
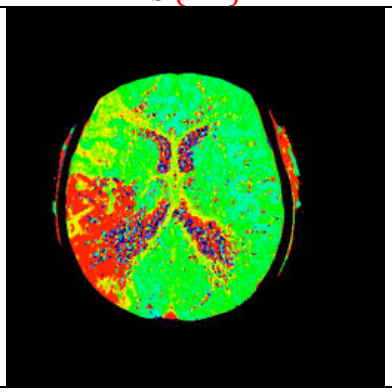
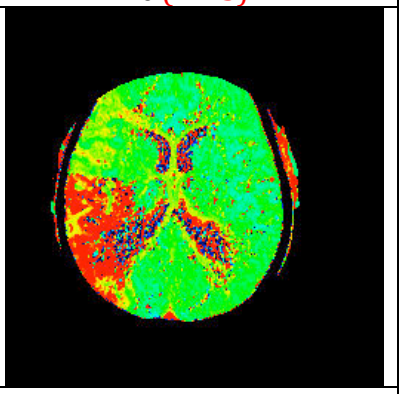
		
a (MEAN)	b (MF)	c (TIPS)
		
d (RAW)	e (GPR)	f (GPR3)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	3.33
b	4.42
c	5.25
d	3.75
e	2.25
f	2

Comments or reasons if they apply:

4. TTP

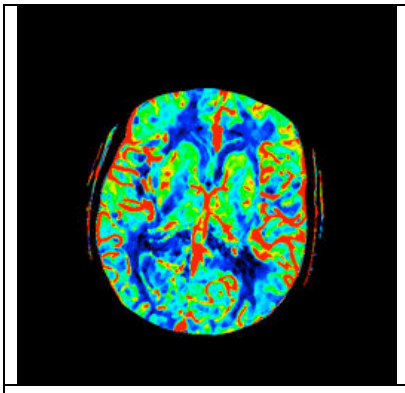
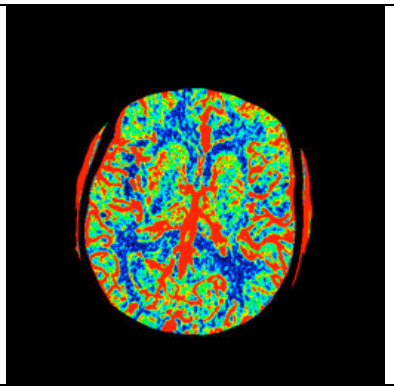
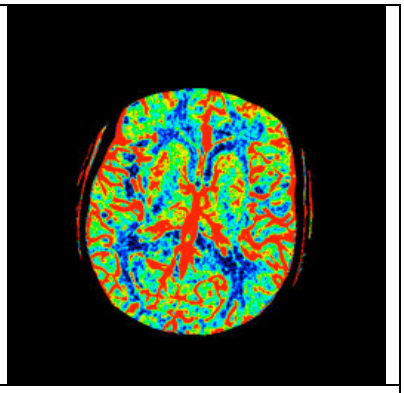
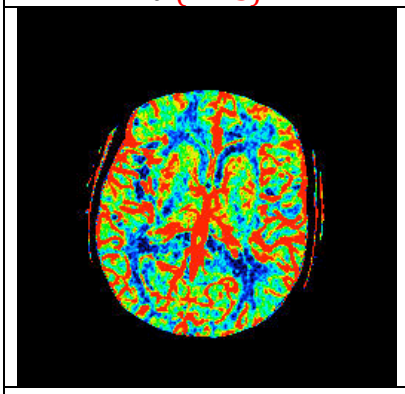
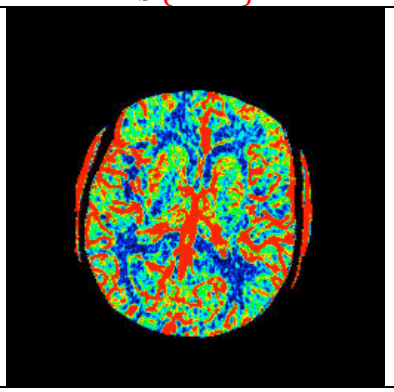
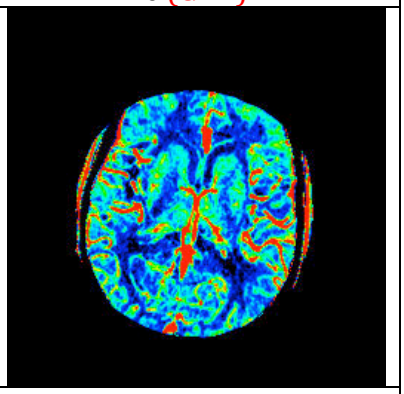
		
a (MEAN)	b (MF)	c (TIPS)
		
d (RAW)	e (GPR)	f (GPR3)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	5.50
b	3.25
c	1.00
d	5.42
e	2.75
f	3.16

Comments or reasons if they apply:

5. CBF

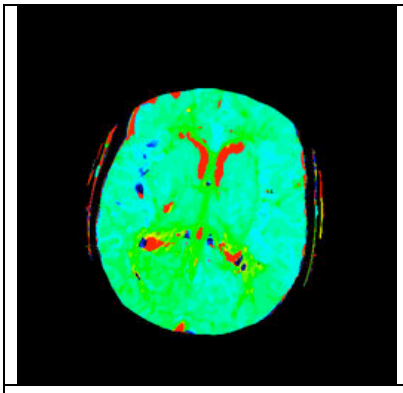
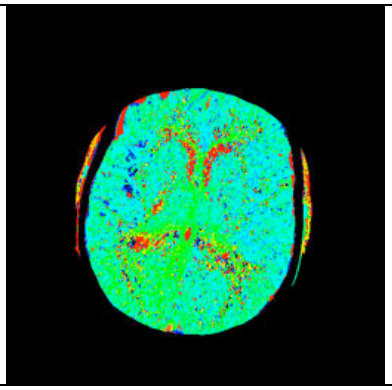
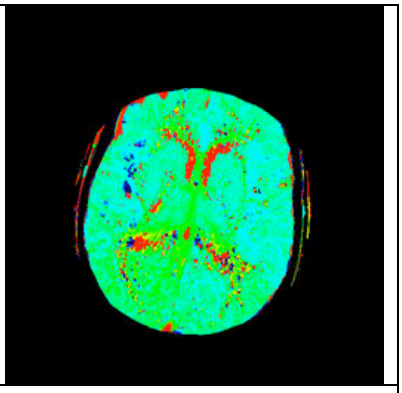
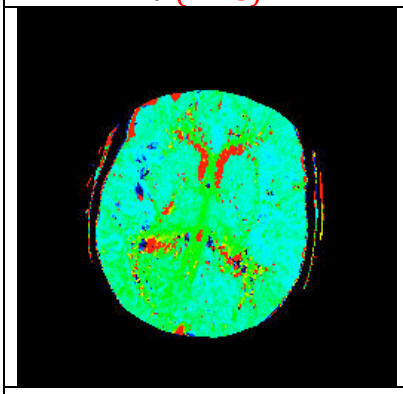
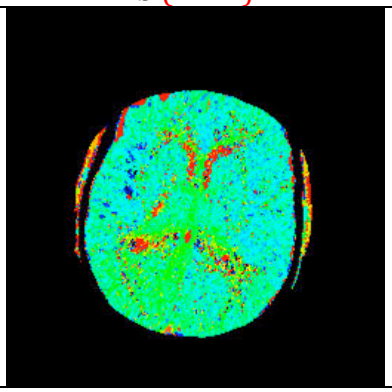
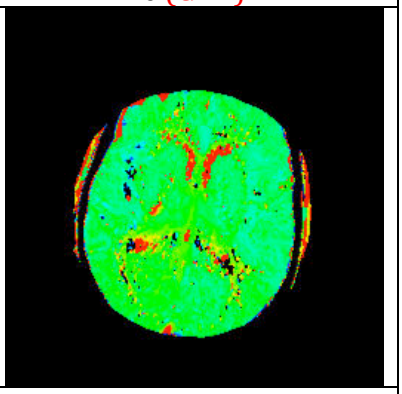
		
a (TIPS)	b (RAW)	c (GPR)
		
d (GPR3)	e (MEAN)	f (MF)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	1.55
b	4.64
c	3.55
d	3.00
e	4.64
f	3.64

Comments or reasons if they apply:

6. TTP

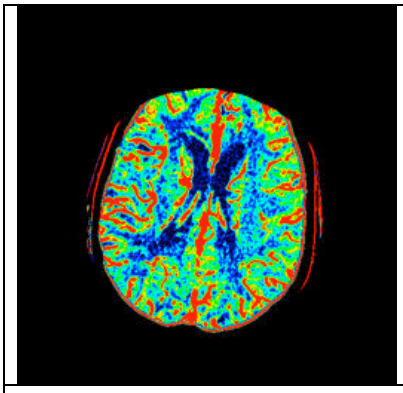
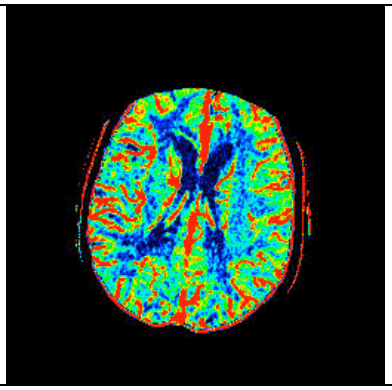
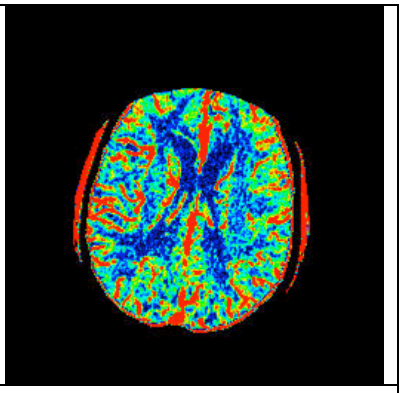
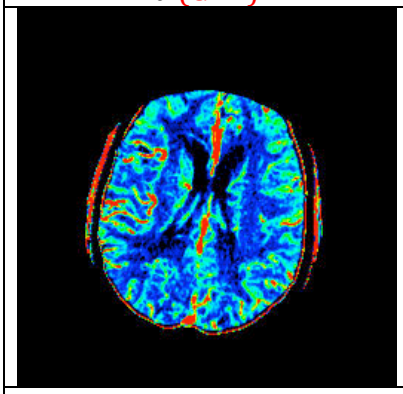
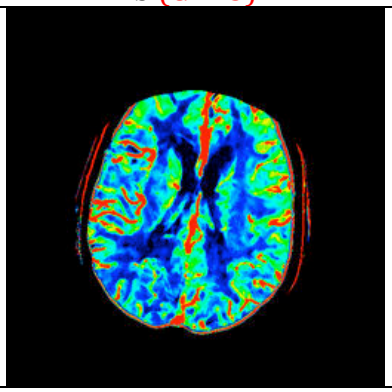
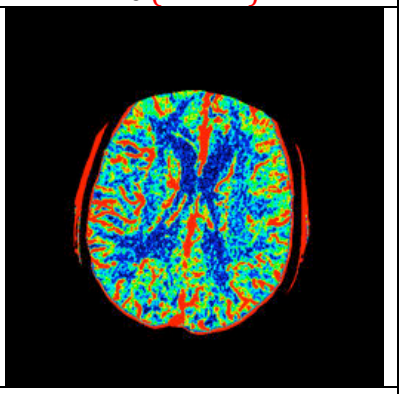
		
a (TIPS)	b (RAW)	c (GPR)
		
d (GPR3)	e (MEAN)	f (MF)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	2.29
b	4.21
c	2.54
d	3.21
e	4.54
f	4.21

Comments or reasons if they apply:

7. CBF

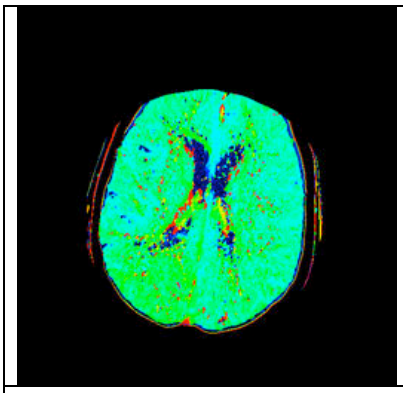
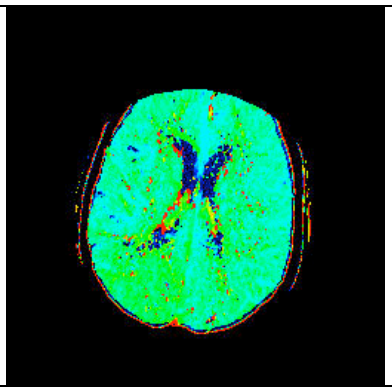
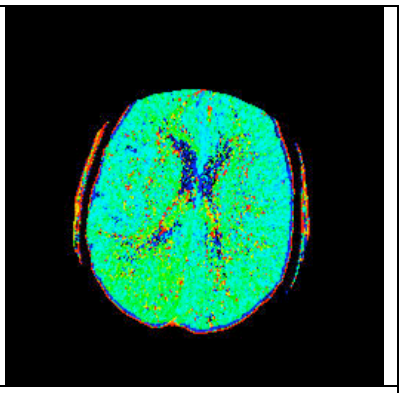
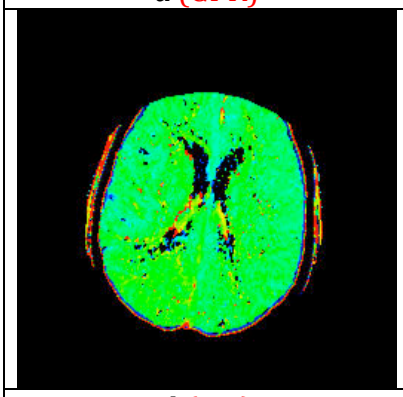
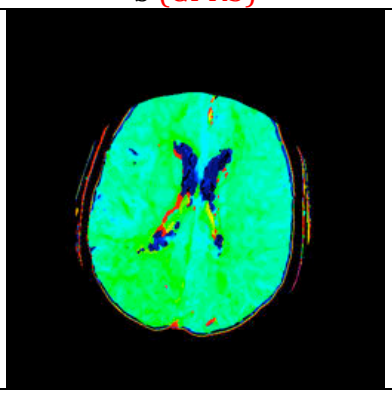
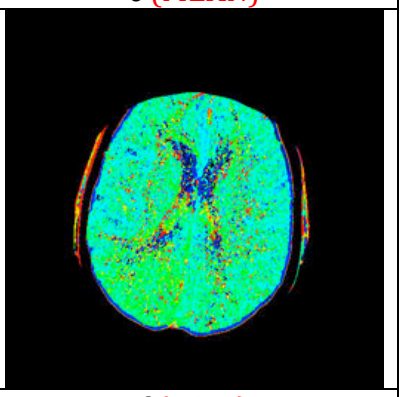
		
a (GPR)	b (GPR3)	c (MEAN)
		
d (MF)	e (TIPS)	f (RAW)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	4.22
b	4.00
c	4.33
d	2.00
e	1.56
f	4.89

Comments or reasons if they apply:

8. TTP

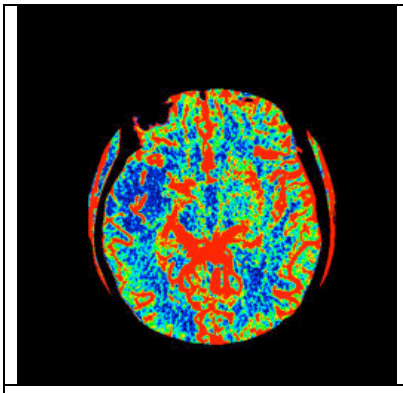
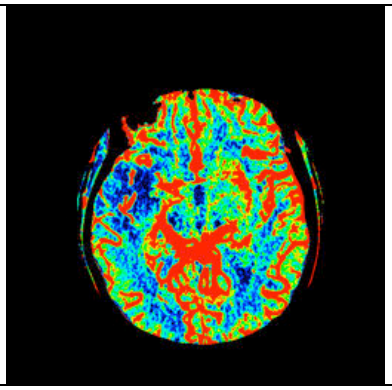
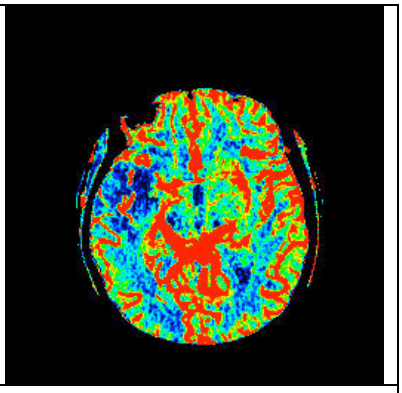
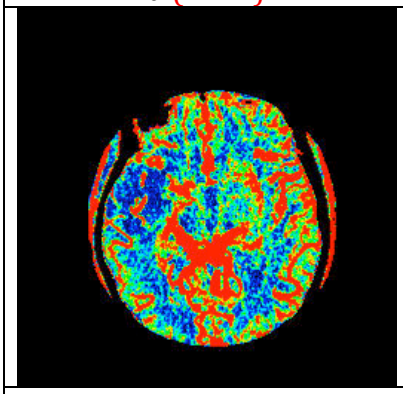
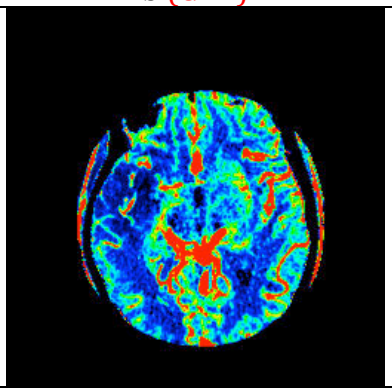
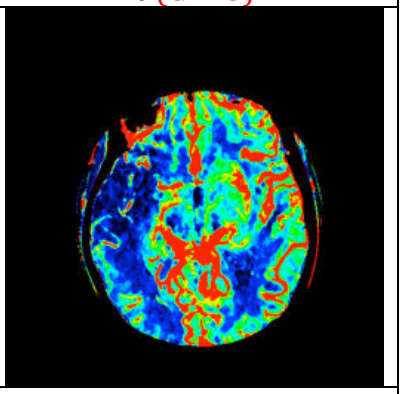
		
a (GPR)	b (GPR3)	c (MEAN)
		
d (MF)	e (TIPS)	f (RAW)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	3.00
b	2.00
c	3.80
d	5.50
e	2.20
f	4.50

Comments or reasons if they apply:

9. CBF

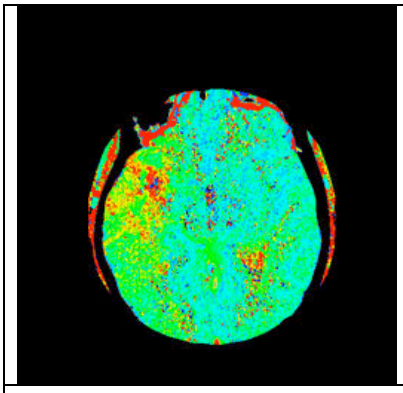
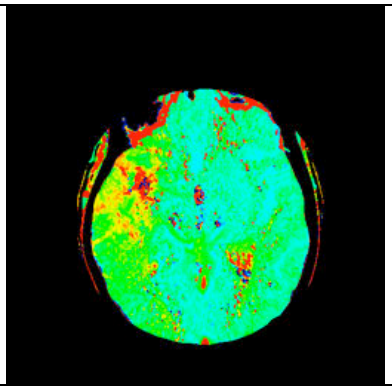
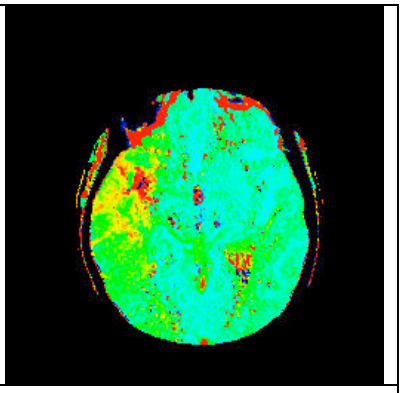
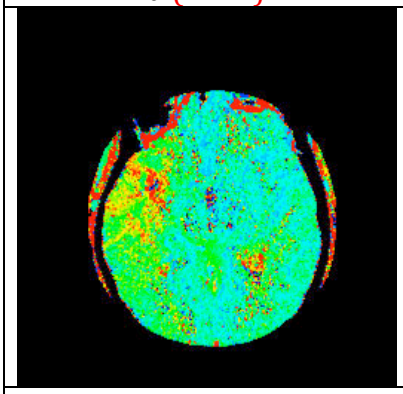
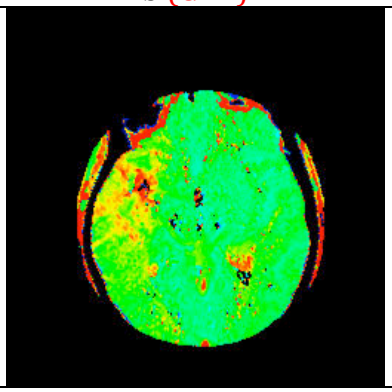
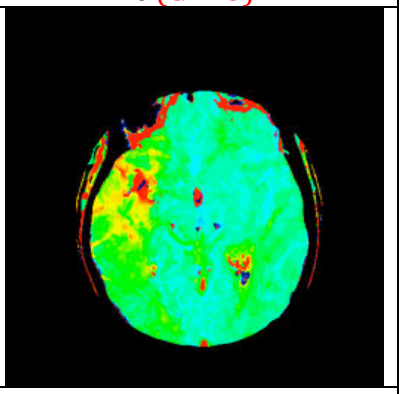
		
a (RAW)	b (GPR)	c (GPR3)
		
d (MEAN)	e (MF)	f (TIPS)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is.

a	3.58
b	3.42
c	3.25
d	4.25
e	3.67
f	2.93

Comments or reasons if they apply:

10. TTP

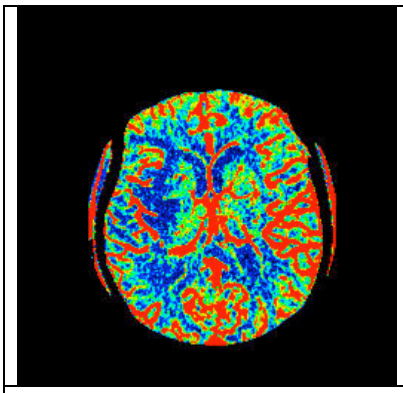
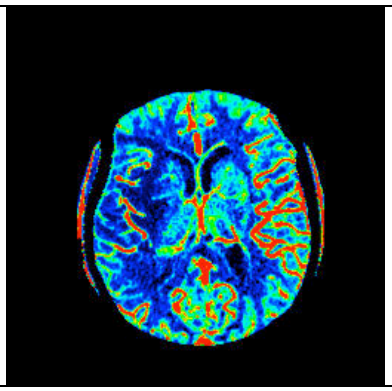
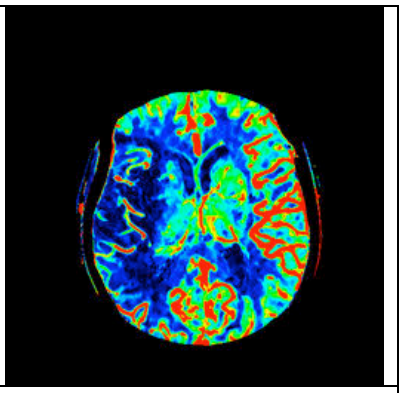
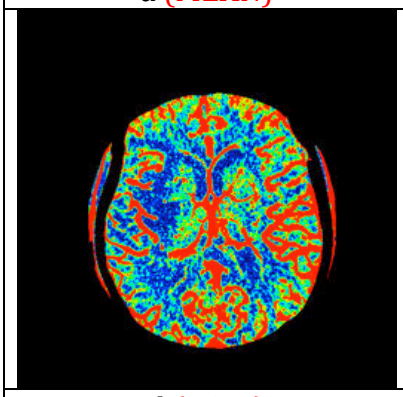
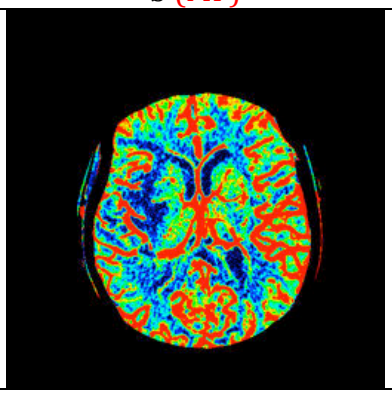
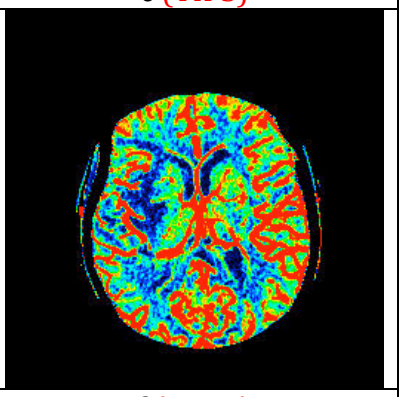
		
a (RAW)	b (GPR)	c (GPR3)
		
d (MEAN)	e (MF)	f (TIPS)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	5.00
b	3.82
c	3.00
d	4.36
e	3.36
f	1.45

Comments or reasons if they apply:

11. CBF

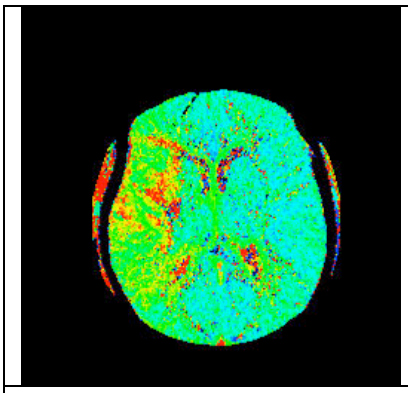
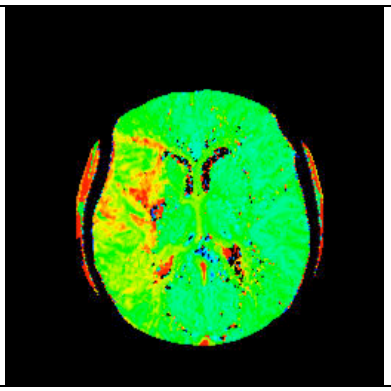
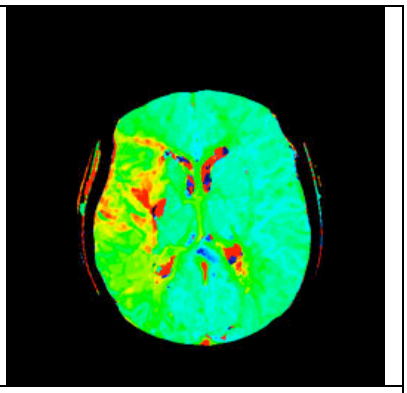
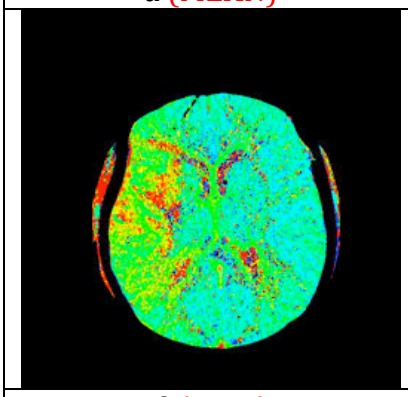
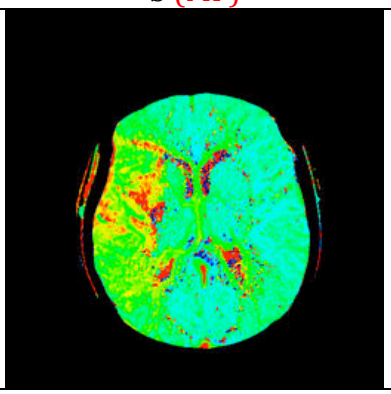
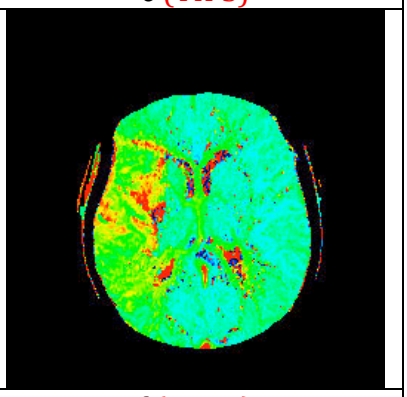
		
a (MEAN)	b (MF)	c (TIPS)
		
d (RAW)	e (GPR)	f (GPR3)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	4.25
b	2.25
c	1.83
d	5.08
e	4.00
f	3.58

Comments or reasons if they apply:

12. TTP

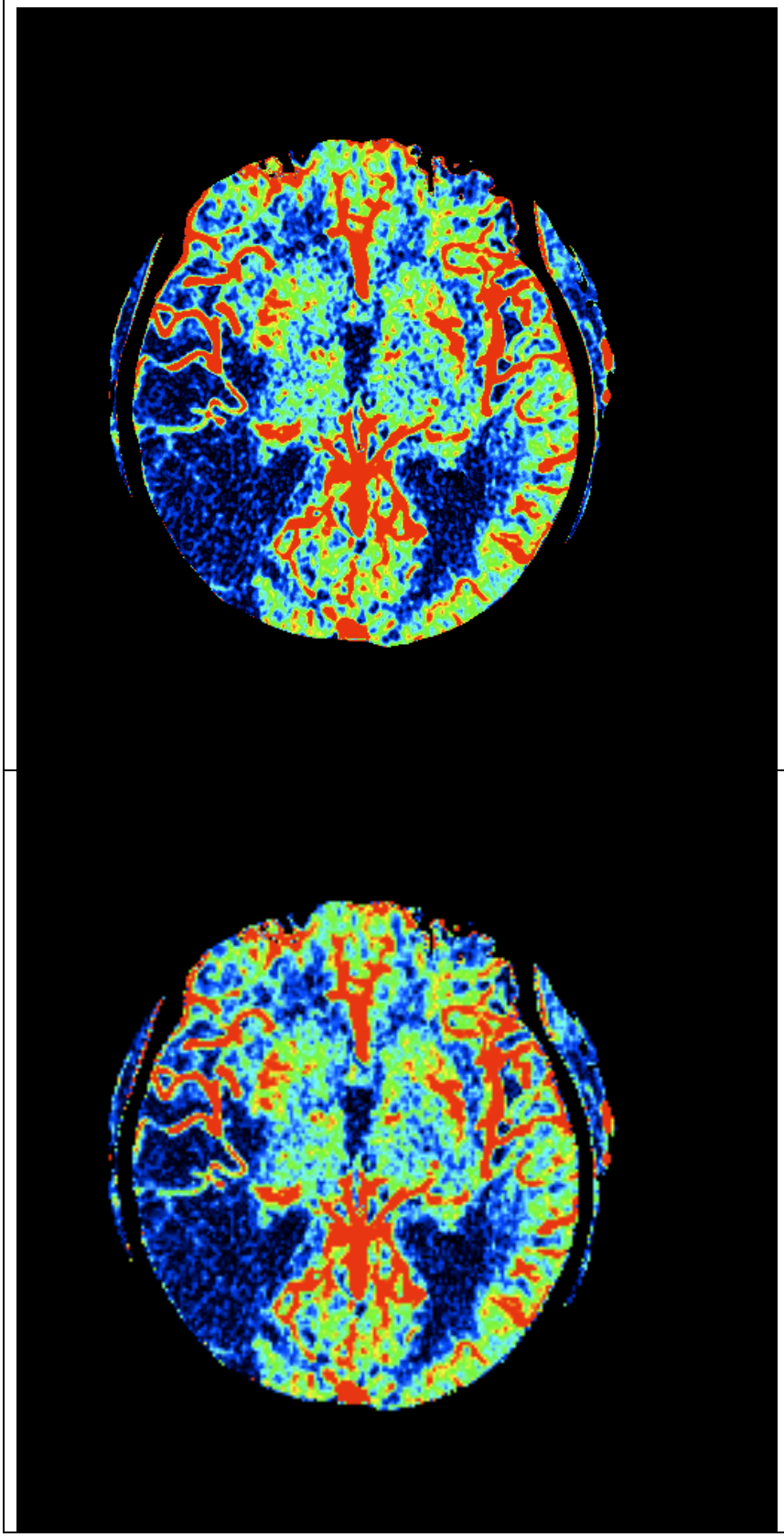
		
a (MEAN)	b (MF)	c (TIPS)
		
d (RAW)	e (GPR)	f (GPR2)

Please put the numbers 1 to 6 against the images where 1 means you think the lesion is clearest.

a	4.67
b	2.67
c	1.33
d	5.08
e	3.50
f	3.75

Comments or reasons if they apply:

13. CBF



How suitably the images would be for diagnostic reporting? Please put the numbers 1 to 10 against the images where 1 means the best and 10 means the worst.

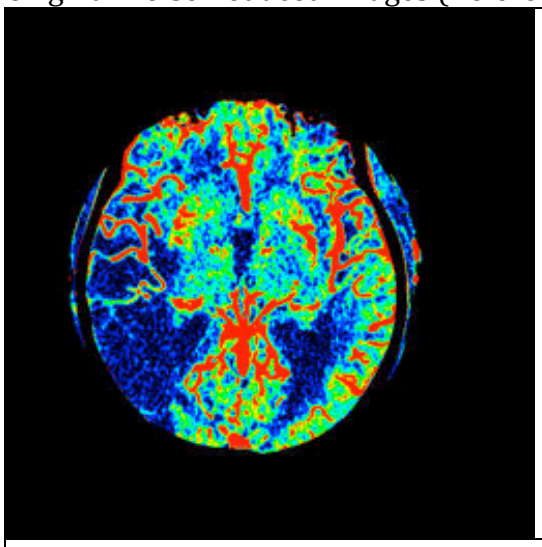
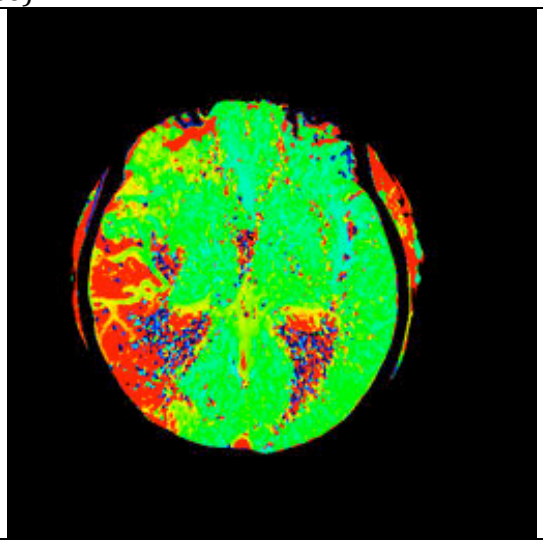
Top: __4.75___

Bottom: __4.58___

14. Automatically Lesion Detection

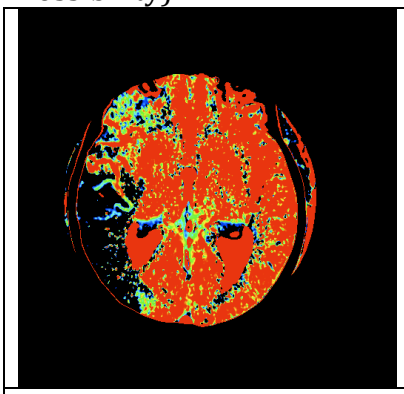
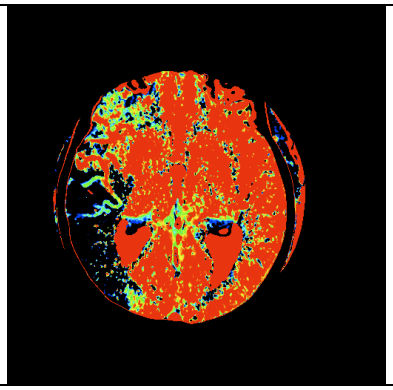
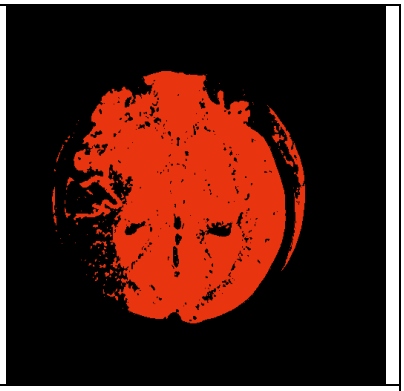
Please tell us how good the auto-segmentation is based on reference CBF and TTP maps.

Original Noise Reduced Images (Reference):

	
CBF	TTP

Segmented Images:

Red= Health; Black = Lesion (High Possibility); Green, Blue = Lesion (Low Possibility)

		
a	b	c

Please put the numbers 1 to 10 against the images where 1 means you think the automatically detected lesion segments are ideal and 10 means they are of no value.

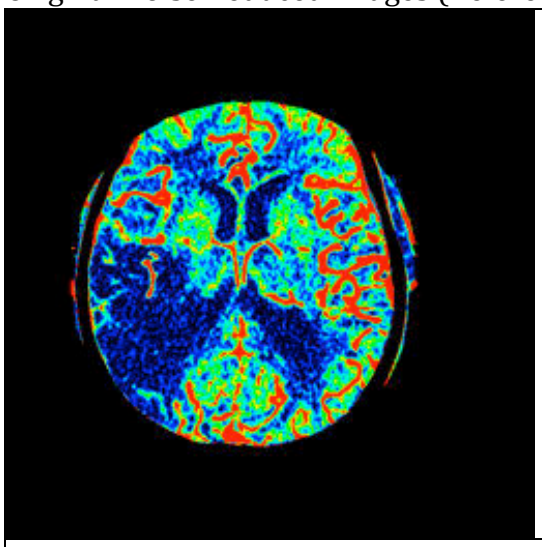
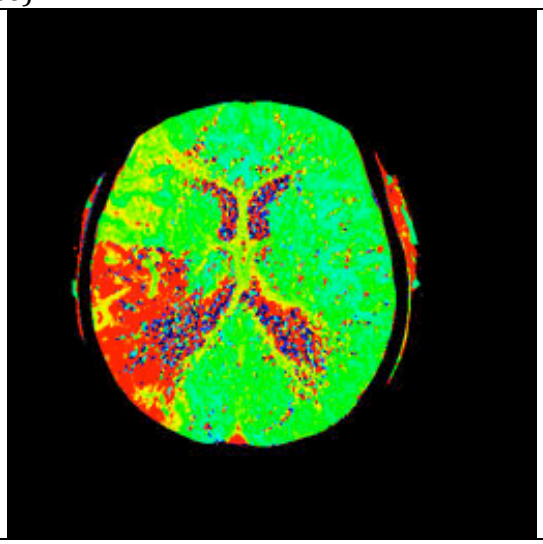
a (Pearson's Correlation)	4.42
b (Spearman's Correlation)	4.67
c (Student t-test)	5.67

Would you find the segmentations above accurate and helpful for clinical decision-making? (For example, do these segmentations spot out some lesion area you may not noticed at the first place?)

15. Automatically Lesion Detection

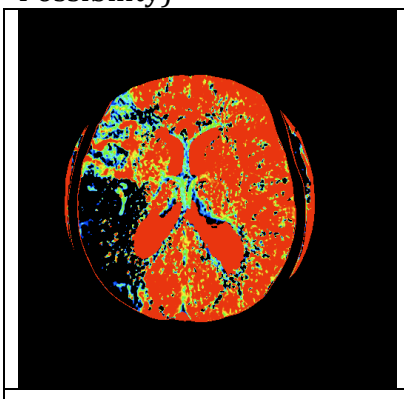
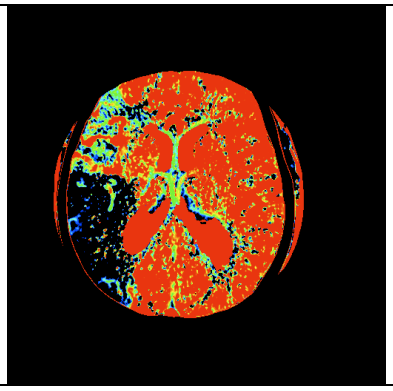
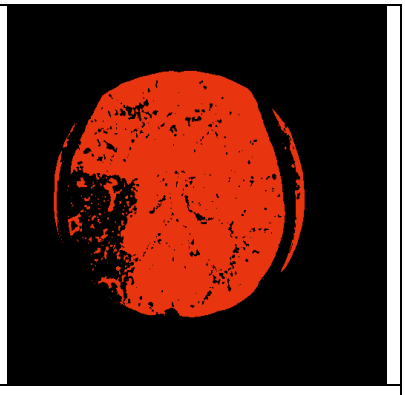
Please tell us how good the auto-segmentation is based on reference CBF and TTP maps

Original Noise Reduced Images (Reference):

	
CBF	TTP

Segmented Images:

Red= Health; Black = Lesion (High Possibility); Green, Blue = Lesion (Low Possibility)

		
a	b	c

Please put the numbers 1 to 10 against the images where 1 means you think the automatically detected lesion segments are ideal and 10 means they are of no value.

a (Pearson's Correlation)	4.08
b (Spearman's Correlation)	4.33
c (Student t-test)	5.75

Would you find the segmentations above accurate and helpful for clinical decision-making?

Bibliography

[1] C. Warlow, M. Dennis, J. Van Gijn, G. Hankey, P. Sandercock, J. Bamford, and J. Wardlaw, *Stroke: A Practical Guide to Management*. Blackwell Publishers, 2001. (Cited on page 2.)

[2] J. Mackay, G. Mensah, S. Mendis, and K. Greenlund, *The atlas of heart disease and stroke*. World Health Organization, 2004. (Cited on page 2.)

[3] The Stroke Association, "www.stroke.org.uk, Accessed May 23, 2012." (Cited on page 2.)

[4] The American Stroke Association, "www.strokeassociation.org, Accessed May 23, 2012." (Cited on page 2.)

[5] The National Stroke Foundation, "strokefoundation.com.au, Accessed May 23, 2012." (Cited on page 2.)

[6] T. Budinger and G. Gullberg, "Three-dimensional reconstruction in nuclear medicine by iterative least-squares and Fourier transform techniques," tech. rep., California Univ., Berkeley (USA). Lawrence Berkeley Lab., 1974. (Cited on page 5.)

[7] L. Shepp and B. Logan, "Reconstructing interior head tissue from x-ray transmissions," *Nuclear Science, IEEE Transactions on*, vol. 21, no. 1, pp. 228–236, 1974. (Cited on page 5.)

[8] P. Sprawls, "AAPM tutoral. CT image detail and noise.," *Radiographics*, vol. 12, no. 5, pp. 1041–1046, 1992. (Cited on pages 5, 20, and 62.)

[9] R. Damadian, "Tumor detection by nuclear magnetic resonance," *Science*, vol. 171, no. 3976, p. 1151, 1971. (Cited on page 5.)

[10] S. Kety and C. Schmidt, "The determination of cerebral blood flow in man by the use of nitrous oxide in low concentrations," *American Journal of Physiology–Legacy Content*, vol. 143, no. 1, p. 53, 1945. (Cited on page 6.)

[11] N. Lassen, "Cerebral blood flow and oxygen consumption in man.," *Physiological reviews*, vol. 39, no. 2, p. 183, 1959. (Cited on page 6.)

[12] L. Østergaard, "Principles of cerebral perfusion imaging by bolus tracking," *Journal of Magnetic Resonance Imaging*, vol. 22, no. 6, pp. 710–717, 2005. (Cited on page 7.)

[13] A. Konstas and M. Lev, "CT perfusion imaging of acute stroke: The need for arrival time, delay insensitive, and standardized postprocessing algorithms? 1," *Radiology*, vol. 254, no. 1, pp. 22–25, 2010. (Cited on page 7.)

[14] B. Rosen, J. Belliveau, J. Vevea, and T. Brady, "Perfusion imaging with NMR contrast agents," *Magnetic resonance in medicine*, vol. 14, no. 2, pp. 249–265, 1990. (Cited on page 7.)

[15] S. Davis, G. Donnan, M. Parsons, C. Levi, K. Butcher, A. Peeters, P. Barber, C. Bladin, D. De Silva, G. Byrnes, *et al.*, "Effects of alteplase beyond 3 h after stroke in the echoplanar imaging thrombolytic evaluation trial (epithet): a placebo-controlled randomised trial," *The Lancet Neurology*, vol. 7, no. 4, pp. 299–309, 2008. (Cited on page 8.)

[16] J. Petrella and J. Provenzale, "MR perfusion imaging of the brain: techniques and applications," *American Journal of Roentgenology*, vol. 175, no. 1, p. 207, 2000. (Cited on page 9.)

[17] Acute Stroke Imaging Standardization Group - Japan (ASIST-Japan), "Perfusion mismatch analyzer (PMA)." software,, Nov 2006. (Cited on pages 9, 57, and 69.)

[18] M. Straka, G. Albers, and R. Bammer, "Real-time diffusion-perfusion mismatch analysis in acute stroke," *Journal of Magnetic Resonance Imaging*, vol. 32, no. 5, pp. 1024–1037, 2010. (Cited on page 10.)

[19] L. Axel, "Cerebral blood flow determination by rapid-sequence computed tomography: theoretical analysis.," *Radiology*, vol. 137, no. 3, pp. 679–686, 1980. (Cited on page 10.)

[20] A. de Gonzalez and S. Darby, "Risk of cancer from diagnostic X-rays: estimates for the UK and 14 other countries," *The Lancet*, vol. 363, no. 9406, pp. 345–351, 2004. (Cited on pages 11 and 56.)

[21] A. Einstein, M. Henzlova, and S. Rajagopalan, "Estimating risk of cancer associated with radiation exposure from 64-slice computed tomography coronary angiography," *JAMA: the journal of the American Medical Association*, vol. 298, no. 3, p. 317, 2007. (Cited on pages 11 and 56.)

[22] M. Wintermark, J. Thiran, P. Maeder, P. Schnyder, and R. Meuli, "Simultaneous measurement of regional cerebral blood flow by perfusion CT and stable xenon CT: a validation study," *American journal of neuroradiology*, vol. 22, no. 5, pp. 905–914, 2001. (Cited on page 12.)

[23] T. Mayer, G. Hamann, J. Baranczyk, B. Rosengarten, E. Klotz, M. Wiesmann, U. Missler, G. Schulte-Altedorneburg, and H. Brueckmann, "Dynamic CT perfusion imaging of acute stroke," *American journal of neuroradiology*, vol. 21, no. 8, pp. 1441–1449, 2000. (Cited on page 12.)

[24] M. Wintermark, M. Sesay, E. Barbier, K. Borbely, W. Dillon, J. Eastwood, T. Glenn, C. Grandin, S. Pedraza, J. Soustiel, *et al.*, "Comparative overview of brain perfusion imaging techniques," *Stroke*, vol. 36, no. 9, p. e83, 2005. (Cited on page 12.)

[25] B. Kjølby, L. Østergaard, and V. Kiselev, "Theoretical model of intravascular paramagnetic tracers effect on tissue relaxation," *Magnetic resonance in medicine*, vol. 56, no. 1, pp. 187–197, 2006. (Cited on page 13.)

[26] L. Ostergaard, R. Weisskoff, D. Chesler, C. Gyldensted, and B. Rosen, "High resolution measurement of cerebral blood flow using intravascular tracer bolus passages. part i: Mathematical approach and statistical analysis," *Magn Reson Med*, vol. 36, no. 5, pp. 715–25, 1996. (Cited on pages 14, 16, 19, 57, 80, and 98.)

[27] L. Ostergaard, A. Sorensen, K. Kwong, R. Weisskoff, C. Gyldensted, and B. Rosen, "High resolution measurement of cerebral blood flow using intravascular tracer bolus passages. part ii: Experimental comparison and preliminary results," *Magn Reson Med*, vol. 36, no. 5, pp. 726–36, 1996. (Cited on pages 14, 16, 57, 80, and 98.)

[28] R. Wirestam, L. Andersson, L. Ostergaard, M. Bolling, J. Aunola, A. Lindgren, B. Geijer, S. Holtås, and F. Ståhlberg, "Assessment of regional cerebral blood flow by dynamic susceptibility contrast MRI using different deconvolution techniques," *Magn Reson Med*, vol. 43, no. 5, pp. 691–700, 2000. (Cited on pages 14, 18, 19, 80, and 98.)

[29] S. Sourbron, R. Luypaert, P. Schuerbeek, M. Dujardin, and T. Stadnik, "Choice of the regularization parameter for perfusion quantification with mri," *Physics in Medicine and Biology*, vol. 49, p. 3307, 2004. (Cited on page 18.)

[30] S. Sourbron, M. Dujardin, S. Makkat, and R. Luypaert, "Pixel-by-pixel deconvolution of bolus-tracking data: optimization and implementation," *Physics in medicine and biology*, vol. 52, p. 429, 2007. (Cited on page 18.)

[31] G. Golub, P. Hansen, and D. O'Leary, "Tikhonov regularization and total least squares," *SIAM Journal on Matrix Analysis and Applications*, vol. 21, no. 1, 1999. (Cited on page 18.)

[32] O. Wu, L. Ostergaard, R. Weisskoff, T. Benner, B. Rosen, and A. Sorensen, "Tracer arrival timing-insensitive technique for estimating flow in MR perfusion-weighted imaging using singular value decomposition with a block-circulant deconvolution matrix," *Magn Reson Med*, vol. 50, no. 1, pp. 164–74, 2003. (Cited on pages 18 and 19.)

[33] I. Andersen, A. Szymkowiak, C. Rasmussen, L. Hanson, J. Marstrand, H. Larsson, and L. Hansen, "Perfusion quantification using Gaussian process deconvolution," *Magn Reson Med*, vol. 48, no. 2, pp. 351–61, 2002. (Cited on page 19.)

[34] F. Calamante, M. Mørup, and L. Hansen, "Defining a local arterial input function for perfusion mri using independent component analysis," *Magnetic resonance in Medicine*, vol. 52, no. 4, pp. 789–797, 2004. (Cited on pages 20 and 24.)

[35] C. Lorenz, T. Benner, P. J. Chen, C. J. Lopez, H. Ay, M. W. Zhu, N. M. Menezes, H. Aronen, J. Karonen, Y. Liu, J. Nuutinen, and A. G. Sorensen, "Automated perfusion-weighted MRI using localized arterial input functions using localized arterial input functions," *J Magn Reson Imaging*, vol. 24, pp. 1133–1139, Nov 2006. (Cited on pages 20, 24, 36, and 52.)

[36] C. Gomez, "Time is brain," *J Stroke Cerebrovasc Dis*, vol. 3, pp. 1–2, 1993. (Cited on page 20.)

[37] J. Saver, "Time is brain - quantified," *Stroke*, vol. 37, no. 1, pp. 263–266, 2006. (Cited on pages 21 and 24.)

[38] F. Zhu, D. R. Gonzalez, T. Carpenter, M. Atkinson, and J. Wardlaw, "A parallel deconvolution algorithm in perfusion imaging," in *Healthcare Informatics, Imaging and Systems Biology (HISB), 2011 First IEEE International Conference on*, pp. 278–283, IEEE, 2011. (Cited on pages 22 and 23.)

[39] F. Zhu, D. Gonzalez, T. Carpenter, M. Atkinson, and J. Wardlaw, "Parallel perfusion imaging processing using GPGPU," *Computer Methods and Programs in Biomedicine*, 2012. (Cited on pages 22 and 23.)

[40] F. Zhu, T. Carpenter, D. R. Gonzalez, M. Atkinson, and J. Wardlaw, "Computed tomography perfusion imaging denoising using gaussian process regression," *Physics in Medicine and Biology*, vol. 57, no. 12, pp. N183–198, 2012. (Cited on pages 22 and 56.)

[41] F. Calamante, D. Gadian, and A. Connelly, "Delay and dispersion effects in dynamic susceptibility contrast MRI: simulations using singular value decomposition," *Magnetic resonance in medicine*, vol. 44, no. 3, pp. 466–473, 2000. (Cited on page 24.)

[42] F. Calamante, D. Gadian, and A. Connelly, "Quantification of perfusion using bolus tracking magnetic resonance imaging in stroke," *Stroke*, vol. 33, no. 4, pp. 1146–1151, 2002. (Cited on page 24.)

[43] G. Duhamel, G. Schlaug, and D. Alsop, "Measurement of arterial input functions for dynamic susceptibility contrast magnetic resonance imaging using echoplanar images: comparison of physical simulations with in vivo results," *Magn Reson Med*, vol. 55, no. 3, pp. 514–23, 2006. (Cited on page 24.)

[44] D. Alsop, A. Wedmid, and G. Schlaug, "Defining a local input function for perfusion quantification with bolus contrast MRI," in *Proceedings of the 10th Annual Meeting of ISMRM, Honolulu*, p. 659, 2002. (Cited on page 24.)

[45] B. Barney, "Introduction to parallel computing, lawrence livermore national laboratory," 2005. (Cited on page 27.)

[46] O. OpenMP, "A proposed industry standard API for shared memory programming," *OpenMP Architecture Review Board*, 1997. (Cited on pages 27 and 49.)

[47] L. Dagum and R. Menon, "OpenMP: an industry standard API for shared-memory programming," *Computational Science & Engineering, IEEE*, vol. 5, no. 1, pp. 46–55, 1998. (Cited on page 27.)

[48] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI: portable parallel programming with the message passing interface*. Scientific And Engineering Computation Series, MIT Press, 1994. (Cited on pages 27 and 49.)

[49] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI, 2nd Edition: portable parallel programming with the message passing interface*. Scientific And Engineering Computation Series, MIT Press, 1999. (Cited on page 27.)

[50] L. Adhianto and B. Chapman, "Performance modeling of communication and computation in hybrid mpi and openmp applications," *Simulation Modelling Practice and Theory*, vol. 15, no. 4, pp. 481–491, 2007. (Cited on page 28.)

[51] NVIDIA Corporation, "Nvidia website, www.nvidia.com, accessed may 23, 2012." (Cited on page 28.)

[52] Current Analysis Inc., "Current analysis website, www.currentanalysis.com, accessed may 23, 2012." (Cited on page 29.)

[53] NVIDIA Corporation, "CUDA programming guide." 2012. (Cited on pages 29 and 37.)

[54] G. Moore *et al.*, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998. (Cited on page 29.)

[55] M. Harris, "GPGPU: General-purpose computation on GPUs," in *Game Developers Conference*, 2005. (Cited on page 30.)

[56] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, *et al.*, "A view of the parallel computing landscape," *Communications of the ACM*, vol. 52, no. 10, pp. 56–67, 2009. (Cited on page 30.)

[57] S. Lahabar and P. J. Narayanan, "Singular value decomposition on GPU using CUDA," *2009 IEEE International Symposium on Parallel and Distributed Processing*, 2009. (Cited on pages 30, 39, and 40.)

[58] N. Govindaraju, B. Lloyd, Y. Dotsenko, B. Smith, and J. Manferdelli, "High performance discrete fourier transforms on graphics processors," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pp. 1–12, Ieee, 2008. (Cited on page 30.)

[59] Z. Yang, Y. Zhu, and Y. Pu, "Parallel image processing based on cuda," in *Computer Science and Software Engineering, 2008 International Conference on*, vol. 3, pp. 198–201, Ieee, 2008. (Cited on page 30.)

[60] F. Vazquez, J. Fern, J. Martinez, and J. Fernandez, "The sparse matrix vector product on GPUs," *aceuales*, pp. 1–13, 2009. (Cited on page 30.)

[61] V. Lee, C. Kim, J. Chhugani, M. Deisher, D. Kim, A. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, *et al.*, "Debunking the 100X GPU vs. CPU myth: an evaluation of throughput computing on CPU and GPU," *ISCA '10 Proceedings of the 37th annual international symposium on Computer architecture*, vol. 38, no. 3, pp. 451–460, 2010. (Cited on page 30.)

[62] E. Lindholm, J. Nickolls, S. Oberman, and J. Montrym, "NVIDIA Tesla: A unified graphics and computing architecture," *Micro, IEEE*, vol. 28, no. 2, pp. 39–55, 2008. (Cited on page 31.)

[63] R. Cox, J. Ashburner, H. Breman, K. Fissell, C. Haselgrove, C. Holmes, J. Lancaster, D. Rex, S. Smith, J. Woodward, *et al.*, "A (sort of) new image data format standard: Nifti-1," *Human Brain Mapping*, vol. 25, 2004. (Cited on pages 33 and 34.)

[64] J. Tesic, "Evaluating a class of dimensionality reduction algorithms.". (Cited on page 35.)

[65] E. Anderson, Z. Bai, and C. Bischof, *LAPACK Users' guide*, vol. 9. Society for Industrial Mathematics, 1999. (Cited on page 39.)

[66] A. Kerr, D. Campbell, and M. Richards, "QR decomposition on GPUs," in *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, pp. 71–78, ACM, 2009. (Cited on page 39.)

[67] "www.cpubenchmark.net," June 20th 2012. (Cited on page 50.)

[68] C. Rasmussen, "Gaussian processes in machine learning," in *Advanced Lectures on Machine Learning* (O. Bousquet, U. von Luxburg, and G. Rätsch, eds.), vol. 3176 of *Lecture Notes in Computer Science*, pp. 63–71, Springer Berlin / Heidelberg, 2004. (Cited on page 56.)

[69] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006. (Cited on pages 56, 63, and 65.)

[70] T. Bronikowski, C. Dawson, and J. Linehan, "Model-free deconvolution techniques for estimating vascular transport functions," *International journal of biomedical computing*, vol. 14, no. 5, pp. 411–429, 1983. (Cited on page 57.)

[71] H. Wittsack, A. Wohlschlager, E. Ritzl, R. Kleiser, M. Cohnen, R. Seitz, and U. Moodder, "CT-perfusion imaging of the human brain: advanced deconvolution analysis using circulant singular value decomposition," *Computerized Medical Imaging and Graphics*, vol. 32, no. 1, pp. 67–77, 2008. (Cited on page 57.)

[72] A. Witkin, *Scale-space filtering*. San Francisco: Morgan Kaufmann, 1987. (Cited on page 57.)

[73] L. Østergaard, D. Smith, P. Vestergaard-Poulsen, S. Hansen, A. Gee, A. Gjedde, and C. Gyldensted, "Absolute cerebral blood flow and blood volume measured by magnetic resonance imaging bolus tracking: comparison with positron emission tomography values," *Journal of Cerebral Blood Flow & Metabolism*, vol. 18, no. 4, pp. 425–432, 1998. (Cited on page 57.)

[74] G. Pajares and J. Manueldelacruz, "A wavelet-based image fusion tutorial," *Pattern Recognition*, vol. 37, no. 9, pp. 1855–1872, 2004. (Cited on pages 57 and 60.)

[75] A. Borsdorf, R. Raupach, and J. Hornegger, "Wavelet based noise reduction by identification of correlations," *Lecture Notes in Computer Science*, vol. 4174, pp. 21–30, 2006. (Cited on page 57.)

[76] A. Borsdorf, R. Raupach, and J. Hornegger, "Separate CT-reconstruction for orientation and position adaptive wavelet denoising," *Informatik aktuell*, vol. 10, pp. 232–236, 2007. (Cited on page 57.)

[77] J.-L. Starck, E. Candes, and D. Donoho, "The curvelet transform for image denoising.," *Image Processing, IEEE Transactions on*, vol. 11, no. 6, pp. 670–684, 2002. (Cited on page 57.)

[78] R. Sivakumar, "Denoising of computer tomography images using curvelet transform," *ARPN Journal of Engineering and Applied Sciences*, vol. 2, no. 1, pp. 21–26, 2007. (Cited on page 57.)

[79] S. A. Hyder and R. Sukanesh, "An efficient algorithm for denoising MR and CT images using digital curvelet transform," *Adv Exp Med Biol*, vol. 696, pp. 471–480, 2011. (Cited on page 57.)

[80] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Computer Vision, 1998. Sixth International Conference on*, pp. 839–846, IEEE, 1998. (Cited on pages 59 and 60.)

[81] A. Grossmann and J. Morlet, "Decomposition of hardy functions into square integrable wavelets of constant shape," *SIAM J. Math. Anal*, vol. 15, pp. 723–736, 1984. (Cited on page 60.)

[82] A. Borsdorf, R. Raupach, T. Flohr, and J. Hornegger, "Wavelet based noise reduction in CT-images using correlation analysis," *Medical Imaging, IEEE Transactions on*, vol. 27, no. 12, pp. 1685 –1703, 2008. (Cited on page 60.)

[83] A. Mendrik, E. Vonken, B. van Ginneken, H. de Jong, A. Riordan, T. van Seeters, E. Smit, M. Viergever, and M. Prokop, "TIPS bilateral noise reduction in 4D CT perfusion scans produces high-quality cerebral blood flow maps," *Physics in Medicine and Biology*, vol. 56, p. 3857, 2011. (Cited on pages 61 and 67.)

[84] B. Whiting, P. Massoumzadeh, O. Earl, J. OSullivan, D. Snyder, and J. Williamson, "Properties of preprocessed sinogram data in x-ray computed tomography," *Medical physics*, vol. 33, p. 3290, 2006. (Cited on page 62.)

[85] M. Ebden, "Gaussian processes for regression: A quick introduction," *http://www.robots.ox.ac.uk*, vol. 33, no. August, pp. 137–157, 2008. (Cited on page 63.)

[86] J. Bushberg, J. Seibert, E. Leidholdt Jr, J. Boone, and E. Goldschmidt Jr, "The essential physics of medical imaging," *Medical Physics*, vol. 30, p. 1936, 2003. (Cited on page 70.)

[87] J. Clarkson and C. Adams, *On definitions of bounded variation for functions of two variables*. PhD thesis, Brown University, 1933. (Cited on page 73.)

[88] X. Wang, P. Gao, Y. Lin, L. Ma, J. Xue, B. Sui, C. Wang, and Y. Wang, "Clinical value of computed tomography perfusion source images in acute stroke," *Neurological research*, vol. 31, no. 10, pp. 1079–1083, 2009. (Cited on page 91.)

[89] X. Wang, P. Gao, J. Xue, G. Liu, and L. Ma, "Identification of infarct core and penumbra in acute stroke using CT perfusion source images," *American Journal of Neuroradiology*, vol. 31, no. 1, p. 34, 2010. (Cited on page 91.)

[90] E. Pepper, M. Parsons, G. Bateman, and C. Levi, "Ct perfusion source images improve identification of early ischaemic change in hyperacute stroke," *Journal of clinical neuroscience*, vol. 13, no. 2, pp. 199–205, 2006. (Cited on page 91.)

[91] J. Rodgers and W. Nicewander, "Thirteen ways to look at the correlation coefficient," *American Statistician*, pp. 59–66, 1988. (Cited on page 94.)

[92] J. Myers and A. Well, *Research design and statistical analysis*, vol. 1. Lawrence Erlbaum, 2003. (Cited on page 94.)

[93] T. Seize, "Student's t-test," *Southern Medical Journal*, vol. 70, no. 11, p. 1299, 1977. (Cited on page 96.)

[94] R. Mankiewicz, *The story of mathematics*. Princeton University Press, 2000. (Cited on page 96.)